

1. C Program to Check if a given Integer is Odd or Even.

Problem Description

The program takes the given integer and checks whether the integer is odd or even.

Problem Solution

1. Take the integer to be checked as input.
2. Find the remainder of the integer by dividing it by 2.
3. Use if,else statement to check whether the remainder is equal to zero or not.
4. Print the output and exit.

Program/Source Code

Here is source code of the C program to check whether a given integer is odd or even. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
#include <stdio.h>

#include <conio.h>

void main()

{

int num, remainder;

printf("Enter an integer : ");

scanf("%d", &num);

remainder = num % 2;

if (remainder == 0)

printf("%d is an even integer\n", num);

else

printf("%d is an odd integer\n", num);

}
```

Runtime Test Cases

Case 1:

Enter an integer : 24

24 is an even integer

Case 2:

Enter an integer : 75

75 is an odd integer

Case 3:

Enter an integer : 0

0 is an even integer

2. C Program to Find the Biggest of 3 Numbers

[« Prev](#)

[Next »](#)

This is a C program to calculate the biggest of 3 numbers.

Problem Description

This program takes the 3 numbers and finds the biggest among all.

Problem Solution

1. Take the three numbers as input.
2. Check the first number if it greater than other two.
3. Repeat the step 2 for other two numbers.
4. Print the number which is greater among all and exit.

advertisement

Program/Source Code

Here is source code of the C program to calculate the biggest of 3 numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to find the biggest of three numbers
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int num1, num2, num3;
9.
10.    printf("Enter the values of num1, num2 and num3\n");
11.    scanf("%d %d %d", &num1, &num2, &num3);
12.    printf("num1 = %d\nnum2 = %d\nnum3 = %d\n", num1, num2, num3);
13.    if (num1 > num2)
14.    {
15.        if (num1 > num3)
16.        {
17.            printf("num1 is the greatest among three \n");
18.        }
19.        else
20.        {
21.            printf("num3 is the greatest among three \n");
22.        }
23.    }
24.    else if (num2 > num3)
25.    {
26.        printf("num2 is the greatest among three \n");
27.    }
28.}
```

Program Explanation

1. Take the three numbers and store it in the variables num1, num2 and num3 respectively.
2. Firstly check if the num1 is greater than num2.
3. If it is, then check if it is greater than num3.
4. If it is, then print the output as “num1 is the greatest among three”.
5. Otherwise print the ouput as “num3 is the greatest among three”.

6. If the num1 is not greater than num2, then check if num2 is greater than num3.

7. If it is, then print the output as “num2 is the greatest among three”.

8. Otherwise print the output as “num3 is the greatest among three”.

advertisement

Runtime Test Cases

Case:1

Enter the values of num1, num2 and num3

6 8 10

num1 = 6 num2 = 8 num3 = 10

num3 is the greatest among three

Case:2

Enter the values of num1, num2 and num3

10 87 99

num1 = 10 num2 = 87 num3 = 99

num3 is the greatest among three

3. C Program to Calculate the Value of sin(x)

[« Prev](#)

[Next »](#)

This is a C Program to calculate the value of sin(x).

Problem Description

This C Program Calculates the Value of sin(x).

Problem Solution

It's a non-differentiable function. Start at zero, then goes up to 1, then back down to 0. But then, instead of going negative, it will just "reflect" about the x-axis. The derivative is 1 and then -1 for every x such that $\sin(x) = 0$ (i.e. 0, 180, 360, 540, 720 ...).

advertisement

Program/Source Code

Here is source code of the C program to Calculate the Value of sin(x). The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the value of sin(x) using the series
 * up to the given accuracy (without using user defined function)
 * also print sin(x) using library function.
 */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void main()
{
    int n, x1;
    float accuracy, term, denominator, x, sinx, sinval;

    printf("Enter the value of x (in degrees) \n");
    scanf("%f", &x);
    x1 = x;
    /* Converting degrees to radians */
    x = x * (3.142 / 180.0);
    sinval = sin(x);
    printf("Enter the accuracy for the result \n");
    scanf("%f", &accuracy);
    term = x;
    sinx = term;
    n = 1;
    do
    {
        denominator = 2 * n * (2 * n + 1);
        term = -term * x * x / denominator;
        sinx = sinx + term;
        n = n + 1;
    } while (accuracy <= fabs(sinval - sinx));
    printf("Sum of the sine series = %f\n", sinx);
    printf("Using Library function sin(%d) = %f\n", x1, sin(x));
}
```

Program Explanation

In this C program, we are reading the number of the terms in a series using ‘x’ variable. To convert degrees to radians the following formula is used

$$\text{Sin}(x) = x * (3.142/180.0).$$

advertisement

Do while loop is used to compute the sum of the sine series. Compute the summation of the value of ‘n’ variable with 1 and multiply the value with 2 and again multiply with the value of ‘n’ variable.

Multiply the value of ‘x’ variable twice with the value of ‘term’ variable and take negation of the value then divide the value by ‘denominator’ variable. Compute the summation of the value of ‘sinx’ variable with the value of ‘term’ variable.

While condition is used to check the value of ‘accuracy’ variable is less than or equal to fabs() function value. If the condition is true, then execute the iteration of the loop. Print the value of sin(x) using printf statement.

advertisement

Runtime Test Cases

```
$ cc pgm14.c -lm
$ a.out
Enter the value of x (in degrees)
60
Enter the accuracy for the result
0.86602540378443864676372317075294
Sum of the sine series      = 0.855862
Using Library function sin(60) = 0.866093

$ a.out
Enter the value of x (in degrees)
45
Enter the accuracy for the result
0.70710678118654752440084436210485
Sum of the sine series      = 0.704723
Using Library function sin(45) = 0.707179
```

4. C Program to Find the Number of Integers Divisible by 5

[« Prev](#)

[Next »](#)

This is a C Program which calculates the number of integers divisible by 5 in the given range.

Problem Description

1. This program takes the range as input and finds the number of integers divisible by 5 in the given range.
2. Also finds the sum of all integers that are divisible by 5 in the given range.

Problem Solution

1. Take the range as input.
2. Find all the integers that gives remainder zero when divided by 5 and print them as output.
3. Add all the integers that are divisible by 5 and print the sum.
4. Also print the count of integers that are divisible by 5.

advertisement

Program/Source Code

Here is source code of the C program to calculate the number of integers divisible by 5. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to find the number of integers divisible by
3. * 5 between the given range num1 and num2, where num1 < num2.
4. *
5. * Also find the sum of all these integer numbers which are divisible
6. * by 5 and display the total.
7. */
8. #include <stdio.h>
9.
10.void main()
11{
12    int i, num1, num2, count = 0, sum = 0;
13.
14    printf("Enter the value of num1 and num2 \n");
15    scanf("%d %d", &num1, &num2);
16    /* Count the number and compute their sum*/
17    printf("Integers divisible by 5 are \n");
18    for (i = num1; i < num2; i++)
19    {
20        if (i % 5 == 0)
21        {
22            printf("%3d,", i);
23            count++;
24            sum = sum + i;
25        }
26    }
27    printf("\n Number of integers divisible by 5 between %d and %d =
28    %d\n", num1, num2, count);
29    printf("Sum of all integers that are divisible by 5 = %d\n", sum);
30}
```

Program Explanation

1. Take the range as input and store it in the variables num1 and num2 respectively.
2. Firstly initialize the variables count and sum to zero.
3. Using the for loop, find all the integers that gives remainder zero when divided by 5 and print them consecutively.
4. Along with this, increment both the variables i.e increment the variable count by 1 and variable sum by the number that is divisible by 5.
5. Print the variables count and sum as output.

advertisement

Runtime Test Cases

Case:1

Enter the value of num1 and num2

12 17

Integers divisible by 5 are

15,

Number of integers divisible by 5 between 12 and 17 = 1

Sum of all integers that are divisible by 5 = 15

Case:2

Enter the value of num1 and num2

1 10

Integers divisible by 5 are

5,10

Number of integers divisible by 5 between 1 and 10 = 2

Sum of all integers that are divisible by 5 = 15

5. C Program to Calculate the Mean, Variance & Standard Deviation

[« Prev](#)

[Next »](#)

This is a C Program to calculate the mean, variance & standard deviation.

Problem Description

This C Program calculates the mean, variance & standard deviation.

Problem Solution

The formula which is used in this program is mean = average of the numbers. variance = (summation(($X_i - \text{average of numbers}$) * ($X_i - \text{average of numbers}$)) / Total no of elements. where i = 1 to N here N is the total no of elements. Standard deviation = Squareroot of the variance.

advertisement

Program/Source Code

Here is source code of the C program to calculate the mean, variance & standard deviation. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to input real numbers and find the mean, variance
 * and standard deviation
 */
#include <stdio.h>
#include <math.h>
#define MAXSIZE 10

void main()
{
    float x[MAXSIZE];
    int i, n;
    float average, variance, std_deviatiion, sum = 0, sum1 = 0;

    printf("Enter the value of N \n");
    scanf("%d", &n);
    printf("Enter %d real numbers \n", n);
    for (i = 0; i < n; i++)
    {
        scanf("%f", &x[i]);
    }
    /* Compute the sum of all elements */
    for (i = 0; i < n; i++)
    {
        sum = sum + x[i];
    }
    average = sum / (float)n;
    /* Compute variance and standard deviation */
    for (i = 0; i < n; i++)
    {
        sum1 = sum1 + pow((x[i] - average), 2);
    }
    variance = sum1 / (float)n;
    std_deviatiion = sqrt(variance);
    printf("Average of all elements = %.2f\n", average);
```

```
    printf("variance of all elements = %.2f\n", variance);
    printf("Standard deviation = %.2f\n", std_deviation);
}
```

Program Explanation

In this C Program, we are reading the number of values using ‘n’ variable. Using for loop we are entering the real numbers to compute the mean, variance and standard deviation of the number.

advertisement

For loop is used to calculate the sum of all elements. Compute the average of the value of ‘sum’ variable by the number of elements present in the ‘n’ variable.

Find the variance and standard deviation of the elements. The following formula is used

Variance = (summation ((X[i] – average of numbers) * (X[i] – average of numbers))) / Total number of elements,

Where i = 1 to N here N is the total number of elements

advertisement

Standard deviation = Squareroot of the variance value.

Runtime Test Cases

```
$ cc pgm23.c -lm
$ a.out
Enter the value of N
5
Enter 5 real numbers
34
88
32
12
10
Average of all elements = 35.20
variance of all elements = 794.56
Standard deviation = 28.19
```

6. C Program to Reverse a Number & Check if it is a Palindrome

[« Prev](#)

[Next »](#)

This is a C Program which reverses a number & checks if it is a palindrome or not.

Problem Description

This C program accepts an integer, reverse it and also checks if it is a palindrome or not.

Problem Solution

1. Take the number which you have to reverse as the input.
2. Obtain its quotient and remainder.
3. Multiply the separate variable with 10 and add the obtained remainder to it.
4. Do step 2 again for the quotient and step 3 for the remainder obtained in step 4.
5. Repeat the process until quotient becomes zero.
6. When it becomes zero, check if the reversed number is equal to original number or not.
7. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to reverse a number & checks it is a palindrome or not. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2.  
3. #include <stdio.h>  
4.  
5. void main()  
6. {  
7.     int num, temp, remainder, reverse = 0;  
8.  
9.     printf("Enter an integer \n");  
10.    scanf("%d", &num);  
11.    /* original number is stored at temp */  
12.    temp = num;  
13.    while (num > 0)  
14.    {  
15.        remainder = num % 10;  
16.        reverse = reverse * 10 + remainder;  
17.        num /= 10;  
18.    }  
19.    printf("Given number is = %d\n", temp);  
20.    printf("Its reverse is = %d\n", reverse);  
21.    if (temp == reverse)  
22.        printf("Number is a palindrome \n");  
23.    else  
24.        printf("Number is not a palindrome \n");  
25.}
```

Program Explanation

1. Take the number which you have to reverse as the input and store it in the variable num.
2. Copy the input number to the another variable temp.
3. Firstly initialize the variable reverse to zero.
4. Obtain the remainder of the input number.

5. Multiply the variable reverse with 10 and add the Obtained remainder to it and store the result in the same variable.
6. Obtain the quotient of the input number and considering this as input number repeat the steps as mentioned above until the obtained quotient becomes zero.
7. When it becomes zero, using if,else statement check whether the reversed number is equal to original number or not.
8. If it is equal, then print the output as “Number is a palindrome”, otherwise print the output as “Number is not a palindrome”.

advertisement

Runtime Test Cases

Case:1

Enter an integer

6789

Given number is = 6789

Its reverse is = 9876

Number is not a palindrome

Case:2

Enter an integer

58085

Given number is = 58085

Its reverse is = 58085

Number is a palindrome

7. C Program to Check if a given Integer is Positive or Negative

[« Prev](#)

[Next »](#)

This is a C program to check whether a given integer is positive or negative.

Problem Description

The program takes the given integer and checks whether the integer is positive or negative.

Problem Solution

1. Take the integer which you want to check as input.
2. Check if it is greater or lesser than zero and print the output accordingly.
3. Exit.

advertisement

Program/Source Code

Here is source code of the C program which checks a given integer is positive or negative. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. #include <stdio.h>
2.
3. void main()
4. {
5.     int number;
6.
7.     printf("Enter a number \n");
8.     scanf("%d", &number);
9.     if (number >= 0)
10.        printf("%d is a positive number \n", number);
11.    else
12.        printf("%d is a negative number \n", number);
13. }
```

Program Explanation

1. Take the integer which you want to check as input and store it in a variable number.
2. Using if,else statements check whether the integer is greater or lesser than zero.
3. If it is greater than or equal to zero, then print the ouput as “it is a positive number”.
4. If it is lesser than zero, then print the ouput as “it is a negative number”.
5. Exit.

advertisement

Runtime Test Cases

Case:1
Enter a number
-10
-10 is a negative number

Case:2
Enter a number
45
45 is a positive number

8. C Program to Calculate Sum & Average of an Array

[« Prev](#)

[Next »](#)

This is a C Program to calculate the sum & average of an array.

Problem Description

We have to write a program in C such that we are reading an array of N elements and then we are going to calculate the sum and average of those N elements and display it to the standard output or screen.

Expected Input and Output

If we are entering 5 elements ($N = 5$), with array element values as 10, 20, 30, 40 and 50 then,

1. **Sum of Elements of the array will be:** $10 + 20 + 30 + 40 + 50 = 150$

2. **Average of Elements of the array will be:** $150 / 5 = 30$

advertisement

Problem Solution

Fundamentally, an array is a data structure containing a collection of values or variables. The simplest type of array is a linear array or one-dimensional array. An array can be defined in C with the following syntax:

int Arr[5] = {10, 20, 30, 40, 50};

/ here 10,20,30,40,50 are the elements at indices 0,1,2,3,4 respectively */*

In this example, array **Arr** is a collection of 5 integers. Each integer can be identified and accessed by its index. The indices of the array start with 0. So, the first element of the array will have index 0, the next will have index 1 and so on. For example, if we have to add 20 to the 2nd element, then the syntax will be:

Arr[1] = Arr[1] + 20; (we can also use **Arr[1] += 20;**)

advertisement

For the solution, we will first construct an array with user-defined length and then, we will find its sum and average.

The formula for the average of the array will be:

average = $\Sigma(\text{elements of the array}) / \text{number of elements in the array}$

Simply, we will add all the elements of the array and then divide it with the number of the elements to find the average.

advertisement

The sequence of steps for the solution will be as follows:

1. Take n, a variable that stores the number of elements of the array.
2. Create an array of size n.
3. Iterate via for loop to take array elements as input, and print them.
4. Iterate via for loop to access each element of array to get the sum of all the elements.
5. The average is calculated by dividing the overall sum to the number of elements in the array.
6. Sum and average are printed on the screen.

Program/Source Code

Here is the source code of the C program to calculate the sum & average of an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

1. /*
2. * C program to read N integers into an array A and
3. * a) Find the sum of all numbers

```

4.  * b) Find the average of all numbers
5.  * Display the results with suitable headings
6.  */
7.
8. #include <stdio.h>
9.
10.int main()
11.
12. int i, num;
13. float total = 0.0, average;
14. printf("Enter the value of N \n");
15. scanf("%d", &num);
16. int array[num];
17.
18. printf("Enter %d numbers (-ve, +ve and zero) \n", num);
19.
20. for (i = 0; i < num; i++)
21. {
22.     scanf("%d", &array[i]);
23. }
24.
25. printf("Input array elements \n");
26.
27. for (i = 0; i < num; i++)
28. {
29.     printf("%+3d\n", array[i]);
30. }
31.
32. /* Summation starts */
33.
34. for (i = 0; i < num; i++)
35. {
36.     total+=array[i];/* this means total=total+array[i]; */
37. }
38.
39. average = total / num;
40.
41. printf("\n Sum of all numbers = %.2f\n", total);
42.
43. printf("\n Average of all input numbers = %.2f\n", average);
44.
45.

```

Program Explanation

1. Take a number **num** as input, which will indicate the number of elements in the array.
2. Create an array of integer with user-defined size.
3. Iterating through for loops (from 0 to N)), take integers as input from the user and print them. These inputs are the elements of the array.
4. Now, start summation by iterating through all the elements and adding numbers to calculate the sum of the array.
5. To calculate the average, the overall sum is divided by the total number of elements in the array.
6. Print the sum and average values calculated.

advertisement

Runtime Test Cases

Here is the runtime output of the C program where the user is reading an array of 5 integers with values 10,20,30,40 and 50 and the program is calculating and displaying the sum and average of the elements of the array.

Enter the value of N

5

Enter 5 numbers (-ve, +ve and zero)

10
20
30
40
50

Input array elements

10
20
30
40
50

Sum of all numbers = 150

Average of all input numbers = 30

9. C Program to Calculate the Value of cos(x)

[« Prev](#)

[Next »](#)

This is a C Program to calculate the value of cos(x).

Problem Description

This C Program calculates the value of cos(x).

Problem Solution

Take input from the user and calculates cos(x) value as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C program to calculate the value of cos(x). The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the value of cos(x) using the series
 * up to the given accuracy (without using user defined function)
 * also print cos(x) using library function.
 */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void main()
{
    int n, x1;
    float accuracy, term, denominator, x, cosx, cosval;

    printf("Enter the value of x (in degrees) \n");
    scanf("%f", &x);
    x1 = x;
    /* Converting degrees to radians */
    x = x * (3.142 / 180.0);
    cosval = cos(x);
    printf("Enter the accuracy for the result \n");
    scanf("%f", &accuracy);
    term = 1;
    cosx = term;
    n = 1;
    do
    {
        denominator = 2 * n * (2 * n - 1);
        term = -term * x * x / denominator;
        cosx = cosx + term;
        n = n + 1;
    } while (accuracy <= fabs(cosval - cosx));
    printf("Sum of the cosine series = %f\n", cosx);
    printf("Using Library function cos(%d) = %f\n", x1, cos(x));
}
```

Program Explanation

In this C program, we are reading the number of the terms in a series using ‘n’ variable. To convert degrees to radians the following formula is used

advertisement

$\text{Cos}(x) = x * (3.142/180.0)$.

Do while loop is used to compute the sum of cosine series. Compute the denominator by multiplying the difference of ‘n’ variable value by 1 with 2 and multiply again with ‘n’ variable value by 2.

Multiply the value of ‘x’ variable twice with the value of ‘term’ variable. Take negation of the value then divide the value by ‘denominator’ variable. Compute the summation of the value of ‘cosx’ variable with the value of ‘term’ variable.

advertisement

While condition is used to check the value of ‘accuracy’ variable is less than or equal to fabs() function value. If the condition is true then the iteration of the loop. Print the value of cos(x) using printf statement.

Runtime Test Cases

```
$ cc pgm15.c -lm
$ a.out
Enter the value of x (in degrees)
60
Enter the accuracy for the result
0.86602
Sum of the cosine series      = 0.451546
Using Library function cos(60) = 0.499882

$ a.out
Enter the value of x (in degrees)
45
Enter the accuracy for the result
0.7071
Sum of the cosine series      = 0.691495
Using Library function cos(45) = 0.707035
```

10. C Program to Calculate the Sum of Odd & Even Numbers

[« Prev](#)

[Next »](#)

This is a C program to find the sum of odd and even numbers from 1 to N.

Problem Description

The program takes the number N and finds the sum of odd and even numbers from 1 to N.

Problem Solution

1. Take the number N upto which we have to find the sum as input.
2. Using for loop take the elements one by one from 1 to N.
3. Using if,else statements separate the element as even or odd.
4. Add the even and odd numbers separately and store it in different variables.
5. Print the sum separately and exit.

advertisement

Program/Source Code

Here is source code of the C program to calculate the sum of odd & even numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. #include <stdio.h>
3.
4. void main()
5. {
6.     int i, num, odd_sum = 0, even_sum = 0;
7.
8.     printf("Enter the value of num\n");
9.     scanf("%d", &num);
10.    for (i = 1; i <= num; i++)
11.    {
12.        if (i % 2 == 0)
13.            even_sum = even_sum + i;
14.        else
15.            odd_sum = odd_sum + i;
16.    }
17.    printf("Sum of all odd numbers = %d\n", odd_sum);
18.    printf("Sum of all even numbers = %d\n", even_sum);
19. }
```

Program Explanation

1. User must first enter the number upto which he/she wants to find the sum and is stored in the variable num.
2. Using for loop take the elements one by one from 1 to num.
3. Use if,else statement for each element to find whether it is odd or even by dividing the element by 2.
4. Initialize the variables odd_sum and even_sum to zero.
5. If the element is even,then increment the variable even_sum with the current element.
6. If the element is odd,then increment the variable odd_sum with the current element.
7. Print the variables odd_sum and even_sum separately and exit.

advertisement

Runtime Test Cases

Case 1:

Enter the value of num

10

Sum of all odd numbers = 25

Sum of all even numbers = 30

Case 2:

Enter the value of num

100

Sum of all odd numbers = 2500

Sum of all even numbers = 2550

11. C Program to Print the Factorial of a given Number

[« Prev](#)

[Next »](#)

This is a C Program to print the factorial of a given number.

Problem Description

This C Program prints the factorial of a given number.

Problem Solution

A factorial is product of all the numbers from 1 to n, where n is the user specified number. This program find the product of all the number from 1 to the user specified number.

advertisement

Program/Source Code

Here is source code of the C program to print the factorial of a given number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the factorial of a given number
 */

#include <stdio.h>
void main()
{
    int i, fact = 1, num;

    printf("Enter the number \n");
    scanf("%d", &num);
    if (num <= 0)
        fact = 1;
    else
    {
        for (i = 1; i <= num; i++)
        {
            fact = fact * i;
        }
    }
    printf("Factorial of %d = %5d\n", num, fact);
}
```

Program Explanation

In this C program, we are reading the integer number using ‘num’ integer variable. A factorial is a product of all the numbers from 1 to n, where n is the user specified number.

advertisement

If condition statement is used to check the value of ‘num’ variable is less than or equal to 0. If the condition is true then it will execute the statement and assign the value of ‘fact’ variable as one. Otherwise, if the condition is false then it will execute the else statement. Using for loop multiply all the numbers from 1 to n and display the factorial of a given number as output.

Runtime Test Cases

```
$ cc pgm79.c
$ a.out
Enter the number
10
Factorial of 10 = 3628800
```

12. C Program to Generate Fibonacci Series

[« Prev](#)

[Next »](#)

This is a C program to generate fibonacci series.

Problem Description

This C Program generates fibonacci series.

Problem Solution

In fibonacci series the first two numbers in the Fibonacci sequence are 0 and 1 and each subsequent number is the sum of the previous two. For example fibonacci series is 0, 1, 1, 2, 3, 5, 8,13, 21.....

advertisement

Program/Source Code

Here is source code of the C program to generate fibonacci series. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to generate Fibonacci Series. Fibonacci Series
 * is 0 1 1 2 3 5 8 13 21 ...
 */
#include <stdio.h>

void main()
{
    int fib1 = 0, fib2 = 1, fib3, limit, count = 0;

    printf("Enter the limit to generate the Fibonacci Series \n");
    scanf("%d", &limit);
    printf("Fibonacci Series is ... \n");
    printf("%d\n", fib1);
    printf("%d\n", fib2);
    count = 2;
    while (count < limit)
    {
        fib3 = fib1 + fib2;
        count++;
        printf("%d\n", fib3);
        fib1 = fib2;
        fib2 = fib3;
    }
}
```

Program Explanation

In this C program, we are reading the limit to generate the Fibonacci series using limit variable. In Fibonacci series the first two numbers in the Fibonacci sequence are 0 and 1 and each subsequent number is the sum of the previous two. For example Fibonacci series is 0, 1, 1, 2, 3, 5, 8, 13, 21.....

advertisement

Initially assign the value of ‘fib1’ variable as 0, the value of ‘fib2’ variable as 1 and the value of ‘count’ variable as 2. While loop is used to check the condition that the value of ‘count’ variable is less than the value of ‘limit’ variable.

If the condition is true then execute the loop. Compute the value of ‘fib1’ variable and the value of ‘fib2’ variable then assign the value to ‘fib3’ variable. Increment the value of ‘count’ variable by 1. Assign the value of ‘fib2’ variable to ‘fib1’ variable and the value of ‘fib3’ variable to ‘fib2’ variable. Print the Fibonacci series using printf statement.

Runtime Test Cases

```
$ cc pgm40.c
$ a.out
Enter the limit to generate the Fibonacci Series
6
Fibonacci Series is ...
0
1
1
2
3
5
```

13. C Program to Find the Volume and Surface Area of Cylinder

[« Prev](#)

[Next »](#)

This is a C Program to find the volume and surface area of cylinder.

Problem Description

This C Program calculates volume and surface area of cylinder.

Problem Solution

The formula used in this program Surface_area = $2 * \pi * r * (r + h)$, Volume = $\pi * r * r * h$ where $\pi = 22/7$.

advertisement

Program/Source Code

Here is source code of the C Program to Find the volume and surface area of cylinder. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Volume and Surface Area of cylinder
 */
#include <stdio.h>
#include <math.h>

int main()
{
    float radius, height;
    float surface_area, volume;

    printf("Enter value for radius and height of a cylinder : \n");
    scanf("%f%f", &radius, &height);
    surface_area = 2 * (22 / 7) * radius * (radius + height);
    volume = (22 / 7) * radius * radius * height;
    printf("Surface area of cylinder is: %.3f", surface_area);
    printf("\n Volume of cylinder is : %.3f", volume);
    return 0;
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the radius and height of a cylinder using ‘radius’ and ‘height’ variables respectively. To find the surface area and volume of a cylinder, the following formulas are used.

$$\text{Surface area} = 2 * (22 / 7) * \text{radius} * (\text{radius} + \text{height})$$

$$\text{Volume} = (22 / 7) * \text{radius} * \text{radius} * \text{height}$$

advertisement

Runtime Test Cases

Output:

```
$ cc pgm29.c -lm
```

\$ a.out

Enter value for radius and height of a cylinder :

15 17

Surface area of cylinder is: 2880.000

Volume of cylinder is : 11475.000

14. C Program to Find the Second Largest & Smallest Elements in an Array

[« Prev](#)

[Next »](#)

This C Program finds second largest & smallest elements in an Array.

Problem Description

The program will implement a one dimensional array and sort the array in descending order. Then it finds the second largest and smallest element in an array and also find the average of these two array elements. Later it checks if the resultant average number is present in a given array. If found, display appropriate message.

Problem Solution

1. Create a one dimensional array and fill its content to its size.
2. Arrange the array elements in the descending order.
3. The second largest number would be the 2nd element of array, whereas the second smallest number would be the last second element of array.
4. Take average of these two numbers.
5. Now, look for this average number in the array.

advertisement

Program/Source Code

Here is source code of the C program to find the second largest & smallest elements in an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C program to accept a list of data items and find the second largest
3.  * and smallest elements in it. Compute the average of both and search
4.  * for the average value if it is present in the array.
5.  * Display appropriate message on successful search.
6. */
7.
8.
9. #include <stdio.h>
10. void main ()
11. {
12.
13.     int number[30];
14.     int i, j, a, n, counter, average;
15.
16.     printf("Enter the value of N\n");
17.     scanf("%d", &n);
18.
19.     printf("Enter the numbers \n");
20.     for (i = 0; i < n; ++i)
21.         scanf("%d", &number[i]);
22.
23.     for (i = 0; i < n; ++i)
24.     {
25.         for (j = i + 1; j < n; ++j)
26.         {
27.             if (number[i] < number[j])
28.             {
29.                 a = number[i];
30.                 number[i] = number[j];
```

```

31.         number[j] = a;
32.     }
33. }
34.
35. }
36.
37. printf("The numbers arranged in descending order are given below \n");
38.
39. for (i = 0; i < n; ++i)
40. {
41.     printf("%d\n", number[i]);
42. }
43.
44. printf("The 2nd largest number is = %d\n", number[1]);
45. printf("The 2nd smallest number is = %d\n", number[n - 2]);
46.
47. average = (number[1] + number[n - 2]) / 2;
48. counter = 0;
49.
50. for (i = 0; i < n; ++i)
51. {
52.     if (average == number[i])
53.     {
54.         ++counter;
55.     }
56. }
57.
58. if (counter == 0 )
59.     printf("The average of %d and %d is = %d is not in the array \n",
60.            number[1], number[n - 2], average);
61.
62. else
63.     printf("The average of %d and %d in array is %d in numbers \n",
64.            number[1], number[n - 2], counter);
65. }

```

Program Explanation

1. Declare an array, a of some fixed capacity, 30.
2. Take the size of the array as input from users.
3. Define all the elements of the array using for loop.
4. Sort the element of the array using Insertion Sort in descending order.
5. The second largest number would be the second element of array (array[1]) and second smallest number would be the second last element of array (array[n-2]).
6. Take the average of these two numbers by adding both and dividing by 2.
7. Using for loop, scan each and every element of the array to check whether the element equals the average.
8. If the average is present in the array, then print appropriate message.
9. Exit

advertisement

Runtime Test Cases

Enter the value of N

4

Enter the numbers

450

340

120

670

The numbers arranged in descending order are given below

670

450

340

120

The 2nd largest number is = 450

The 2nd smallest number is = 340

The average of 450 and 340 is = 395 is not in the array

15. C Program to Display the Inventory of Items in a Store

[« Prev](#)

[Next »](#)

This is a C Program to display the inventory of items in a store.

Problem Description

This C Program display the inventory of items in a store.

Problem Solution

The program accepts the value of item name, item code, price, quantity & manufacture date. Then display those value in a structured way.

advertisement

Program/Source Code

Here is source code of the C program to display the inventory of items in a storage. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to display the inventory of items in a store / shop
 * The inventory maintains details such as name, price, quantity
 * and manufacturing date of each item.
 */
#include <stdio.h>

void main()
{
    struct date
    {
        int day;
        int month;
        int year;
    };
    struct details
    {
        char name[20];
        int price;
        int code;
        int qty;
        struct date mfg;
    };
    struct details item[50];
    int n, i;

    printf("Enter number of items:");
    scanf("%d", &n);
    fflush(stdin);
    for (i = 0; i < n; i++)
    {
        fflush(stdin);
        printf("Item name: \n");
        scanf("%s", item[i].name);
        fflush(stdin);
        printf("Item code: \n");
        scanf("%d", &item[i].code);
        printf("Item price: \n");
        scanf("%d", &item[i].price);
        printf("Item quantity: \n");
        scanf("%d", &item[i].qty);
        item[i].mfg.day = 1;
        item[i].mfg.month = 1;
        item[i].mfg.year = 2018;
    }
}
```

```

scanf("%d", &item[i].code);
fflush(stdin);
printf("Quantity: \n");
scanf("%d", &item[i].qty);
fflush(stdin);
printf("price: \n");
scanf("%d", &item[i].price);
fflush(stdin);
printf("Manufacturing date(dd-mm-yyyy): \n");
scanf("%d-%d-%d", &item[i].mfg.day,
&item[i].mfg.month, &item[i].mfg.year);
}
printf("***** INVENTORY ***** \n");
printf("-----\n");
printf("S.N. | NAME | CODE | QUANTITY | PRICE\n");
printf(" | MFG.DATE \n");
printf("-----\n");
for (i = 0; i < n; i++)
printf("%d %15s %d %5d %5d\n",
i + 1, item[i].name, item[i].code, item[i].qty,
item[i].price, item[i].mfg.day, item[i].mfg.month,
item[i].mfg.year);
printf("-----\n");
}

```

Program Explanation

In this C program, integer variables are stored in the structure and the variable item[50] is used to access the integer variable stored in the structure. We are reading the number of variables using ‘n’ variable. The fflush(stdin) function will flush the input buffer of a stream.

advertisement

Using for loop enter the name of the item using ‘item[i].name’ variable, the code of the item using the ‘item[i].code’ variable, the price of the item using the ‘item[i].price’ variable, and the manufacturing date of the item using ‘item[i].mfg.day’, ‘item[i].mfg.month’, ‘item[i].mfg.year’ variables. Then print the values in a structured way.

Runtime Test Cases

```

$ cc pgm60.c
$ a.out
Enter number of items:3
Item name:
pendrive
Item code:
123
Quantity:
6
price:
3000
Manufacturing date(dd-mm-yyyy):
30-9-2012
Item name:
computer
Item code:
124
Quantity:
10
price:

```

10000
Manufacturing date(dd-mm-yyyy):
30-7-2012
Item name:
optical mouse
Item code:
Quantity:
price:
Manufacturing date(dd-mm-yyyy):
***** INVENTORY *****

S.N.	NAME	CODE	QUANTITY	PRICE	MFG.DATE
1	pendrive	123	6	3000	30/9/2012
2	computer	124	10	10000	30/7/2012
3	optical	0	0	0	0/0/0

\$ a.out
Enter number of items:3
Item name:
pendrive
Item code:
123
Quantity:
6
price:
3000
Manufacturing date(dd-mm-yyyy):
30-9-2012

Item name:
computer
Item code:
124
Quantity:
10
price:
10000
Manufacturing date(dd-mm-yyyy):
30-7-2012

Item name:
Mouse
Item code:
125
Quantity:
10
price:
1500
Manufacturing date(dd-mm-yyyy):
30-6-2012

***** INVENTORY *****

S.N.	NAME	CODE	QUANTITY	PRICE	MFG.DATE
1	pendrive	123	6	3000	30/9/2012
2	computer	124	10	10000	30/7/2012
3	Mouse	125	10	1500	30/6/2012

16. C Program to Put Even & Odd Elements of an Array in 2 Separate Arrays

[« Prev](#)

[Next »](#)

This is a C Program to put even & odd elements of an array in 2 separate arrays.

Problem Description

The program first finds the odd and even elements of the array. Then the odd elements of an array is stored in one array and even elements of an array is stored in another array.

Problem Solution

1. Create three integer array. First to store all the elements. Second to store even elements of first array. Third to store odd elements of first array.
2. Take size of the array as input from users and define the first array by inserting elements of the array.
3. Using for loop, scan each and every element of the first array, check whether they are even or odd by taking modulo of 2 on those numbers.
4. Even numbers will get stored(copied) in second array, while odd numbers will get stored in third array.
5. Print both arrays.

advertisement

Program/Source Code

Here is source code of the C program to put even & odd elements of an array in 2 separate arrays. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to accept N integer number and store them in an array AR.
3.  * The odd elements in the AR are copied into OAR and other elements
4.  * are copied into EAR. Display the contents of OAR and EAR.
5. */
6.
7.
8. #include <stdio.h>
9. void main()
10. {
11.
12.     long int ARR[10], OAR[10], EAR[10];
13.     int i, j = 0, k = 0, n;
14.
15.     printf("Enter the size of array AR n");
16.     scanf("%d", &n);
17.
18.     printf("Enter the elements of the array n");
19.     for (i = 0; i < n; i++)
20.     {
21.         scanf("%ld", &ARR[i]);
22.         fflush(stdin);
23.     }
24.
25.     /* Copy odd and even elements into their respective arrays */
26.
27.     for (i = 0; i < n; i++)
28.     {
29.         if (ARR[i] % 2 == 0)
30.         {
```

```

31.     EAR[j] = ARR[i];
32.     j++;
33.   }
34. else
35. {
36.     OAR[k] = ARR[i];
37.     k++;
38.   }
39. }
40.
41. printf("The elements of OAR are n");
42. for (i = 0; i < k; i++)
43. {
44.     printf("%ldn", OAR[i]);
45. }
46.
47. printf("The elements of EAR are n");
48. for (i = 0; i < j; i++)
49. {
50.     printf("%ldn", EAR[i]);
51. }
52.
53. }

```

Program Explanation

1. Create three integer arrays namely ARR, OAR, EAR of some fixed capacity, 10.
2. Take size of the array as input from users and define the first array, ARR by inserting elements of the array.
3. Using for loop, scan each and every element of the first array, check whether they are even or odd by taking modulo of 2 on those numbers.
4. If the array element modulo 2, returns 0, this means the number is divisible by 2 and it is an even number. Add this number to second array.
5. If the array element modulo 2, returns 1, this means the number is not divisible by 2 and it is an odd number. Add this number to third array.
6. Repeat steps 4 and 5 until all the elements of first array are checked.
7. At last, we will be having arrays second third holding all the even and odd elements of first array respectively.
8. Print both the OAR AND EAR.
9. Exit.

advertisement

Runtime Test Cases

Enter the size of array AR

6

Enter the elements of the array

34

56

78

90

12

39

The elements of OAR are

39

The elements of EAR are

34

56

78

90

12

17. C Program to Reverse a Given Number

[« Prev](#)

This is a C program to reverse a given integer.

Problem Description

This C program accepts an integer and reverse it.

Problem Solution

1. Take the number which you have to reverse as the input.
2. Obtain its quotient and remainder.
3. Multiply the separate variable with 10 and add the obtained remainder to it.
4. Do step 2 again for the quotient and step 3 for the remainder obtained in step 4.
5. Repeat the process until quotient becomes zero.
6. When it becomes zero, print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to reverse a given number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. #include <stdio.h>
3.
4. void main()
5. {
6.     long num, reverse = 0, temp, remainder;
7.
8.     printf("Enter the number\n");
9.     scanf("%ld", &num);
10.    temp = num;
11.    while (num > 0)
12.    {
13.        remainder = num % 10;
14.        reverse = reverse * 10 + remainder;
15.        num /= 10;
16.    }
17.    printf("Given number = %ld\n", temp);
18.    printf("Its reverse is = %ld\n", reverse);
19.}
```

Program Explanation

1. Take the number which you have to reverse as the input and store it in the variable num.
2. Copy the input number to the another variable temp.
3. Firstly initialize the variable reverse to zero.
4. Obtain the remainder of the input number.
5. Multiply the variable reverse with 10 and add the Obtained remainder to it and store the result in the same variable.
6. Obtain the quotient of the input number and considering this as input number repeat the steps as mentioned above until the obtained quotient becomes zero.
7. When it becomes zero, print the given number and its reverse using variables temp and reverse respectively as ouput.

advertisement

Runtime Test Cases

Case:1
Enter the number
567865
Given number = 567865
Its reverse is = 568765

Case:2
Enter the number
00001
Given number = 00001
Its reverse is = 10000

18. C Program to Calculate the Area of a Triangle

This is a C Program to calculate the area of a triangle.

Problem Description

This C Program calculates the area of a triangle given it's three sides.

Problem Solution

The formula or algorithm used is: $\text{Area} = \sqrt{s(s - a)(s - b)(s - c)}$, where $s = (a + b + c) / 2$ or perimeter / 2. and a, b & c are the sides of triangle.

advertisement

Program/Source Code

Here is source code of the C program to calculate the area of a triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the area of a triangle, given three sides
 */
#include <stdio.h>
#include <math.h>

void main()
{
    int s, a, b, c, area;

    printf("Enter the values of a, b and c \n");
    scanf("%d %d %d", &a, &b, &c);
    /* compute s */
    s = (a + b + c) / 2;
    area = sqrt(s * (s - a) * (s - b) * (s - c));
    printf("Area of a triangle = %d \n", area);
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the three sides of a triangle using ‘a’, ‘b’, ‘c’ integer variables. To find the area of a triangle, the following formula is used.

$$\text{Area} = \sqrt{s * (s - a) * (s - b) * (s - c)}$$

advertisement

Runtime Test Cases

```
$ cc pgm1.c -lm
$ a.out
Enter the values of a, b and c
12 10 8
Area of a triangle = 39
```

19. C Program to Evaluate the given Polynomial Equation

[« Prev](#)

[Next »](#)

This is a C Program to evaluate the given polynomial equation.

Problem Description

This C Program evaluates the given polynomial equation.

Problem Solution

The polynomial equation formula is $P(x) = A_n X^n + A_{n-1} X^{n-1} + A_{n-2} X^{n-2} + \dots + A_1 X + A_0$.

advertisement

Program/Source Code

Here is source code of the C program to evaluate the given polynomial equation. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to evaluate a given polynomial by reading its coefficients
 * in an array.
 * P(x) = AnXn + An-1Xn-1 + An-2Xn-2+... +A1X + A0
 *
 * The polynomial can be written as:
 * P(x) = A0 + X(A1 + X(A2 + X(A3 + X(Q4 + X(..X(An-1 + XAn))))))
 * and evaluated starting from the inner loop
 */
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 10

void main()
{
    int array[MAXSIZE];
    int i, num, power;
    float x, polySum;

    printf("Enter the order of the polynomial \n");
    scanf("%d", &num);
    printf("Enter the value of x \n");
    scanf("%f", &x);
    /* Read the coefficients into an array */
    printf("Enter %d coefficients \n", num + 1);
    for (i = 0; i <= num; i++)
    {
        scanf("%d", &array[i]);
    }
    polySum = array[0];
    for (i = 1; i <= num; i++)
    {
        polySum = polySum * x + array[i];
    }
    power = num;

    printf("Given polynomial is: \n");
```

```

for (i = 0; i <= num; i++)
{
    if (power < 0)
    {
        break;
    }
    /* printing proper polynomial function */
    if (array[i] > 0)
        printf(" + ");
    else if (array[i] < 0)
        printf(" - ");
    else
        printf(" ");
    printf("%dx^%d ", abs(array[i]), power--);
}
printf("\n Sum of the polynomial = %6.2f\n", polySum);
}

```

Program Explanation

In this C program, we are reading the order of an array using ‘num’ variable and also the value of ‘x’ variable to multiply along with the coefficients. Enter the coefficients of the polynomial equation into an array using for loop.

advertisement

Initially assign the value of ‘array[0]’ variable to ‘polysum’ variable. Using for loop multiply the value of ‘polysum’ variable with the value of ‘x’ variable and assign the coefficient values of array[i] variable to the ‘polysum’ variable.

After evaluating the output, using If-else condition statement display the coefficients in proper polynomial function. If condition statement is used to check the value of coefficient variables is greater than 0. If the condition is true, then it will display a polynomial equation using ‘+’.

Otherwise, if the condition is false then it will execute the elseif condition statement and display the polynomial equation using ‘-’. Again if the condition statement is false, then it will execute the else statement and display the absolute value of the array.

advertisement

Runtime Test Cases

```

$ cc pgm.c
$ a.out
Enter the order of the polynomial
2
Enter the value of x
2
Enter 3 coefficients
3
2
6
Given polynomial is:
+ 3x^2 + 2x^1 + 6x^0
Sum of the polynomial = 22.00

```

```

$ a.out
Enter the order of the polynomial
4
Enter the value of x
1
Enter 5 coefficients

```

3

-5

6

8

-9

Given polynomial is:

$+ 3x^4 - 5x^3 + 6x^2 + 8x^1 - 9x^0$

Sum of the polynomial = 3.00

20. C Program to Find Prime Numbers in a given Range

[« Prev](#)

[Next »](#)

This is a C program to find prime numbers in a given range.

Problem Description

The program takes the range and finds all the prime numbers between the range and also prints the number of prime numbers.

Problem Solution

1. Take the range of numbers between which you have to find the prime numbers as input.
2. Check for prime numbers only on the odd numbers between the range.
3. Also check if the odd numbers are divisible by any of the natural numbers starting from 2.
4. Print the prime numbers and its count.
5. Exit.

advertisement

Program/Source Code

Here is source code of the C program to calculate the prime numbers in a given range. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. void main()
5. {
6.     int num1, num2, i, j, flag, temp, count = 0;
7.
8.     printf("Enter the value of num1 and num2 \n");
9.     scanf("%d %d", &num1, &num2);
10.    if (num2 < 2)
11.    {
12.        printf("There are no primes upto %d\n", num2);
13.        exit(0);
14.    }
15.    printf("Prime numbers are \n");
16.    temp = num1;
17.    if ( num1 % 2 == 0 )
18.    {
19.        num1++;
20.    }
21.    for (i = num1; i <= num2; i = i + 2)
22.    {
23.        flag = 0;
24.        for (j = 2; j <= i / 2; j++)
25.        {
26.            if ((i % j) == 0)
27.            {
28.                flag = 1;
29.                break;
30.            }
31.        }
32.        if(flag == 0)
33.        {
34.            printf("%d\n", i);
```

```
35.     count++;
36. }
37. }
38. printf("Number of primes between %d & %d = %d\n", temp, num2, count);
39. }
```

Program Explanation

1. User must take the range as input and it is stored in the variables num1 and num2 respectively.
2. Initially check whether num2 is lesser than number 2. If it is, then print the output as “there are no prime numbers”.
3. If it is not, then check whether num1 is even. If it is even, then make it odd by incrementing the num1 by 1.
4. Using for loop starting from num1 to num2, check whether the current number is divisible by any of the natural numbers starting from 2. Use another for loop to do this. Increment the first for loop by 2, so as to check only the odd numbers.
5. Firstly initialize the variables flag and count to zero.
6. Use the variable flag to differentiate the prime and non-prime numbers and use the variable count to count the number of prime numbers between the range.
7. Print the prime numbers and variable count separately as output.

advertisement

Runtime Test Cases

Case:1

Enter the value of num1 and num2

70 85

Prime numbers are

71

73

79

83

Number of primes between 70 and 85 = 4

Case:2

Enter the value of num1 and num2

0 1

There are no primes upto 1

21. C Program to read two Strings & Concatenate the Strings

[« Prev](#)

[Next »](#)

This is a C program to read two strings & concatenate the strings.

Problem Description

This program takes two strings as input and concatenate them.

Problem Solution

1. Take two strings as input and store them in two different arrays.
2. Find the position of the last element of the first array and from that position keep on adding the elements of the second array.
3. Exit.

advertisement

Program/Source Code

Here is source code of the C program to read two strings & concatenate the strings. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2.  
3. /*  
4.  * C program to read two strings and concatenate them, without using  
5.  * library functions. Display the concatenated string.  
6. */  
7. #include <stdio.h>  
8. #include <string.h>  
9.  
10.void main()  
11.{  
12.    char string1[20], string2[20];  
13.    int i, j, pos;  
14.  
15.    /* Initialize the string to NULL values */  
16.    memset(string1, 0, 20);  
17.    memset(string2, 0, 20);  
18.  
19.    printf("Enter the first string : ");  
20.    scanf("%s", string1);  
21.    printf("Enter the second string: ");  
22.    scanf("%s", string2);  
23.    printf("First string = %s\n", string1);  
24.    printf("Second string = %s\n", string2);  
25.  
26.    /* Concat the second string to the end of the first string */  
27.    for (i = 0; string1[i] != '\0'; i++)  
28.    {  
29.        /* null statement: simply traversing the string1 */  
30.        ;  
31.    }  
32.    pos = i;  
33.    for (j = 0; string2[j] != '\0'; i++)  
34.    {  
35.        string1[i] = string2[j++];  
36.    }
```

```
37. /* set the last character of string1 to NULL */
38. string1[i] = '\0';
39. printf("Concatenated string = %s\n", string1);
40. }
```

Program Explanation

1. Take two strings as input and store them in the arrays string1 and string2 respectively.
2. Using for loop find the position of the last element of the array string1[]. Store that position in the variable pos.
3. Using another for loop add the elements of the array string2[] into the array string1[] starting from the obtained position.
4. Print the array string1[] as output.

advertisement

Runtime Test Cases

```
Enter the first string : San
Enter the second string: foundry
First string = San
Second string = foundry
Concatenated string = Sanfoundry
```

22. C Program to Calculate the Area of a Circle

[Next »](#)

This is a C Program to calculate the area of a circle.

Problem Description

This C Program calculates the area of a Circle given it's radius.

Problem Solution

The formula to calculate the area is: $\text{Area} = \pi \times r^2$ where r is the radius of the circle & π value is 22/7.

advertisement

Program/Source Code

Here is source code of the C program to calculate the area of a circle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the area of a circle, given the radius
 */
#include <stdio.h>
#include <math.h>
#define PI 3.142

void main()
{
    float radius, area;

    printf("Enter the radius of a circle \n");
    scanf("%f", &radius);
    area = PI * pow(radius, 2);
    printf("Area of a circle = %5.2f\n", area);
}
```

Program Explanation

In this C program, library function defined in `<math.h>` header file is used to compute mathematical functions. We are reading the radius of a circle using 'radius' variable. To find the area of a circle, the following formula is used.

$$\text{Area} = \text{PI} * \text{pow}(\text{radius}, 2)$$

advertisement

Runtime Test Cases

```
$ cc pgm2.c -lm
$ a.out
Enter the radius of a circle
30
Area of a circle = 2827.80
```

23. C Program to Calculate the Sum of the Elements of each Row & Column

[« Prev](#)

[Next »](#)

This is a C program to calculate the sum of the elements of each row & column in a given matrix.

Problem Description

This C program reads a matrix A (MxN) & finds the following using functions

- a) Sum of the elements of each row
- b) Sum of the elements of each column
- c) Find the sum of all the elements of the matrix

Problem Solution

1. Take the MxN matrix as input.
2. Define two functions separately for row sum and column sum.
3. Use for loops to calculate the sum of the elements of each row & column in a given matrix.
4. Either add all the calculated row sum or column sum to get the sum of all elements of the matrix.

advertisement

Program/Source Code

Here is source code of the C program to calculate the sum of the elements of each row & column. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to read a matrix A (MxN) & find the following using
3.  * functions a) Sum of the elements of each row
4.  * b) Sum of the elements of each column
5.  * c) Find the sum of all the elements of the matrix
6.  * Output the computed results
7. */
8. #include <stdio.h>
9. int Addrow(int array1[10][10], int k, int c);
10.int Addcol(int array1[10][10], int k, int r);
11.
12.void main()
13.{
14.    int arr[10][10];
15.    int i, j, row, col, rowsum, colsum, sumall=0;
16.
17.    printf("Enter the order of the matrix \n");
18.    scanf("%d %d", &row, &col);
19.    printf("Enter the elements of the matrix \n");
20.    for (i = 0; i < row; i++)
21.    {
22.        for (j = 0; j < col; j++)
23.        {
24.            scanf("%d", &arr[i][j]);
25.        }
26.    }
27.    printf("Input matrix is \n");
28.    for (i = 0; i < row; i++)
29.    {
30.        for (j = 0; j < col; j++)
31.        {
32.            printf("%3d", arr[i][j]);
33.        }
34.    }
35.}
```

```

33.    }
34.    printf("\n");
35. }
36. /* computing row sum */
37. for (i = 0; i < row; i++)
38. {
39.     rowsum = Addrow(arr, i, col);
40.     printf("Sum of row %d = %d\n", i + 1, rowsum);
41. }
42. /* computing col sum */
43. for (j = 0; j < col; j++)
44. {
45.     colsum = Addcol(arr, j, row);
46.     printf("Sum of column %d = %d\n", j + 1, colsum);
47. }
48. /* computation of all elements */
49. for (j = 0; j < row; j++)
50. {
51.     sumall = sumall + Addrow(arr, j, col);
52. }
53. printf("Sum of all elements of matrix = %d\n", sumall);
54. }

55./* Function to add each row */
56.int Addrow(int array1[10][10], int k, int c)
57.{
58.    int rsum = 0, i;
59.    for (i = 0; i < c; i++)
60.    {
61.        rsum = rsum + array1[k][i];
62.    }
63.    return(rsum);
64.}

65./* Function to add each column */
66.int Addcol(int array1[10][10], int k, int r)
67.{
68.    int csum = 0, j;
69.    for (j = 0; j < r; j++)
70.    {
71.        csum = csum + array1[j][k];
72.    }
73.    return(csum);
74.}

```

Program Explanation

- Take M & N of a MxN matrix as input and store it in the variables row & column respectively.
- Take all the elements of the matrix using two for loops and store in the array a[][].
- Define two functions namely Addrow and Addcol for calculating row sum and column sum respectively.
- Call each function inside a for loop.
- Use another for loop inside each function to calculate their respective sum.
- Use variables rsum & csum to store row sum and column sum respectively.
- Print the output using variables rsum and csum.
- For the sum of all elements of the matrix call either function Addrow or Addcol inside a for loop.
- Firstly initialize the variable sumall to zero, later increment it with the returned values from the called function.
- Print the variable sumall and exit.

advertisement

Runtime Test Cases

Enter the order of the matrix

3 3

Enter the elements of the matrix

2 3 4

7 1 5

3 8 9

Input matrix is

2 3 4

7 1 5

3 8 9

Sum of row 1 = 9

Sum of row 2 = 13

Sum of row 3 = 20

Sum of column 1 = 12

Sum of column 2 = 12

Sum of column 3 = 18

Sum of all elements of matrix = 42

24. C Program to Find if a given Year is a Leap Year

[« Prev](#)

[Next »](#)

This is a C program to find whether a given year is leap year or not.

Problem Description

This program takes a year as input and finds whether a year is leap year or not.

Problem Solution

1. Take a year as input.
2. Check whether a given year is divisible by 400.
3. Check whether a given year is divisible by 100.
4. Check whether a given year is divisible by 4.
5. If the condition at step 2 and 4 becomes true, then the year is a leap year.
6. If the condition at step 3 becomes true, then the year is not a leap year.

advertisement

Program/Source Code

Here is source code of the C program to check a given year is leap year. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to find whether a given year is leap year or not
3. */
4. void main()
5. {
6.     int year;
7.
8.     printf("Enter a year \n");
9.     scanf("%d", &year);
10.    if ((year % 400) == 0)
11.        printf("%d is a leap year \n", year);
12.    else if ((year % 100) == 0)
13.        printf("%d is a not leap year \n", year);
14.    else if ((year % 4) == 0)
15.        printf("%d is a leap year \n", year);
16.    else
17.        printf("%d is not a leap year \n", year);
18.}
```

Program Explanation

1. Take a year as input and store it in the variable year.
2. Using if,else statements to,
 - a) Check whether a given year is divisible by 400.
 - b) Check whether a given year is divisible by 100.
 - c) Check whether a given year is divisible by 4.
3. If the condition at step 2.a becomes true, then print the ouput as “It is a leap year”.
4. If the condition at step 2.b becomes true, then print the ouput as “It is not a leap year”.
5. If the condition at step 2.c becomes true, then print the ouput as “It is a leap year”.
6. If neither of the condition becomes true, then the year is not a leap year and print the same.

advertisement

Runtime Test Cases

Enter a year

2012

2012 is a leap year

Enter a year

2009

2009 is not a leap year

25. C Program to Convert a Decimal Number to Binary & Count the Number of 1s

[« Prev](#)

[Next »](#)

This is a C Program to convert a decimal number to binary & count the number of 1s.

Problem Description

This C Program converts a decimal number into binary & count the number of 1s.

Problem Solution

The program uses module operation and multiplication with base 2 operation for conversion. It also uses modulo operation to check for 1's and accordingly increments the count of 1s.

advertisement

Program/Source Code

Here is source code of the C program to convert a decimal number to binary & count the number of 1s. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to accept a decimal number and convert it to binary
 * and count the number of 1's in the binary number
 */
#include <stdio.h>

void main()
{
    long num, decimal_num, remainder, base = 1, binary = 0, no_of_1s = 0;

    printf("Enter a decimal integer \n");
    scanf("%ld", &num);
    decimal_num = num;
    while (num > 0)
    {
        remainder = num % 2;
        /* To count no.of 1s */
        if (remainder == 1)
        {
            no_of_1s++;
        }
        binary = binary + remainder * base;
        num = num / 2;
        base = base * 10;
    }
    printf("Input number is = %d\n", decimal_num);
    printf("Its binary equivalent is = %ld\n", binary);
    printf("No.of 1's in the binary number is = %d\n", no_of_1s);
}
```

Program Explanation

In this C program, we are reading the decimal number using ‘num’ variable. A decimal number system is a base 10 number system using digits for 0 to 9 whereas binary number system is base 2 and uses 0 and 1. Check

whether the number is less than or equal to zero. Divide the number by 2 and store the remainder in the array. Increase the length of the array by 1. After the execution of while loop, print the binary number and the number of 1's.

advertisement

Runtime Test Cases

```
$ cc pgm46.c
$ a.out
Enter a decimal integer
134
Input number is = 134
Its binary equivalent is = 10000110
No.of 1's in the binary number is = 3
```

26. C Program to Swap the Contents of two Numbers using Bitwise XOR Operation

[« Prev](#)

[Next »](#)

This is a C Program to swap the contents of two numbers using bitwise xor operation.

Problem Description

This C Program swaps the contents of two numbers using bitwise XOR operation.

Problem Solution

Take input from the user and swaps the contents of two numbers using bitwise XOR operation.

advertisement

Program/Source Code

Here is source code of the C program to swap the contents of two numbers using bitwise XOR operation. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to swap the contents of two numbers using bitwise XOR
 * operation. Don't use either the temporary variable or arithmetic
 * operators
 */
#include <stdio.h>

void main()
{
    long i, k;

    printf("Enter two integers \n");
    scanf("%ld %ld", &i, &k);
    printf("\n Before swapping i= %ld and k = %ld", i, k);
    i = i ^ k;
    k = i ^ k;
    i = i ^ k;
    printf("\n After swapping i= %ld and k = %ld", i, k);
}
```

Program Explanation

In this C program, we are reading two integers using ‘i’ and ‘k’ integer variables respectively. To swap the integer values without using a temporary variable or arithmetic operators. The bitwise XOR operator is used to copy the bit if it is set in one operand but not both. Print the swapped values of numbers.

advertisement

Runtime Test Cases

```
$ cc pgm48.c
```

```
$ a.out
```

```
Enter two integers
```

45

89

Before swapping i= 45 and k = 89

After swapping i= 89 and k = 45

27. C Program to Convert a Given Number of Days in terms of Years, Weeks & Days

[« Prev](#)

[Next »](#)

This a C program which converts a given number of Days in terms of years, weeks & days.

Problem Description

This program takes the number of days as input and converts in terms of years, weeks & days.

Problem Solution

1. Take the number of days as input.
2. For the number of years, divide the input by 365 and obtain its quotient.
3. For the number of weeks, divide the input by 365 and obtain its remainder.Further divide the remainder by 7(no of days in a week) and obtain its quotient.
4. For the number of days, divide the input by 365 and obtain its remainder.Further divide the remainder by 7(no of days in a week) and obtain its remainder.

advertisement

Program/Source Code

Here is source code of the C program to converts a given number of Days in terms of years, weeks & days. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to convert given number of days to a measure of time given
3. * in years, weeks and days. For example 375 days is equal to 1 year
4. * 1 week and 3 days (ignore leap year)
5. */
6. #include <stdio.h>
7. #define DAYSINWEEK 7
8.
9. void main()
10. {
11.     int ndays, year, week, days;
12.
13.     printf("Enter the number of days\n");
14.     scanf("%d", &ndays);
15.     year = ndays / 365;
16.     week =(ndays % 365) / DAYSINWEEK;
17.     days =(ndays % 365) % DAYSINWEEK;
18.     printf ("%d is equivalent to %d years, %d weeks and %d daysn",
19.             ndays, year, week, days);
20. }
```

Program Explanation

1. Take the number of days as input and store it in variable ndays.
2. For the number of years, divide the input by 365(no of days in a year) and obtain its quotient.Store this in the variable year.
3. For the number of weeks, divide the input by 365 and obtain its remainder.Further divide the remainder by 7(no of days in a week) and obtain its quotient.Store this in the variable week.
4. For the number of days, divide the input by 365 and obtain its remainder.Further divide the remainder by 7(no of days in a week) and obtain its remainder.Store this in the variable days.
5. Print the output and exit.

Runtime Test Cases

Case:1

Enter the number of days

29

29 is equivalent to 0 years, 4 weeks and 1 days

Case:2

Enter the number of days

1000

1000 is equivalent to 2 years, 38 weeks and 4 days

28. C Program to Calculate the Simple Interest

[« Prev](#)

[Next »](#)

This is a C Program to calculate the simple interest.

Problem Description

This C Program calculates the simple interest.

Problem Solution

This C Program calculates the simple interest given the principal amount, rate of interest and time. The formula to calculate the simple interest is: $\text{simple_interest} = (\text{p} * \text{t} * \text{r}) / 100$ where p is principal amount, t is time & r is rate of interest.

advertisement

Program/Source Code

Here is source code of the C program to calculate the simple interest. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the simple interest, given principal,
 * rate of interest and time.
 */
#include <stdio.h>

void main()
{
    float principal_amt, rate, simple_interest;
    int time;

    printf("Enter the values of principal_amt, rate and time \n");
    scanf("%f %f %d", &principal_amt, &rate, &time);
    simple_interest = (principal_amt * rate * time) / 100.0;
    printf("Amount = Rs. %.2f\n", principal_amt);
    printf("Rate = Rs. %.2f%\n", rate);
    printf("Time = %d years\n", time);
    printf("Simple interest = %.2f\n", simple_interest);
}
```

Program Explanation

This C program, we are reading the value for principal_amt, rate and time using the ‘principal_amt’, ‘rate’ and ‘time’ variables respectively.

advertisement

To find the simple interest, the following formula is used.

$$\text{simple_interest} = (\text{principal_amt} * \text{time} * \text{rate}) / 100$$

Runtime Test Cases

```
$ cc pgm3.c
$ a.out
Enter the values of principal_amt, rate and time
```

12

10

5

Amount = Rs. 12.00

Rate = Rs. 10.00%

Time = 5 years

Simple interest = 6.00

29. C Program to Accepts two Strings & Compare them

[« Prev](#)

[Next »](#)

This is C program which accepts two strings & compare them.

Problem Description

This program accepts two strings as input and compares them.

Problem Solution

1. Take two strings as input.
2. Compare the two strings and display the result whether both are equal, or first string is greater than the second or the first string is less than the second string
3. Exit.

advertisement

Program/Source Code

Here is source code of the C program to accepts two strings & compare them. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. #include <stdio.h>
3.
4. int main ()
5. {
6.     int count1 = 0, count2 = 0, flag = 0, i;
7.     char string1[30], string2[30];
8.
9.     printf ("Enter the First string\n");
10.    gets (string1);
11.
12.    printf ("Enter the Second string\n");
13.    gets (string2);
14.
15.    while (string1[count1] != '\0')
16.        count1++;
17.
18.    while (string2[count2] != '\0')
19.        count2++;
20.
21.    i = 0;
22.
23.    while (string1[i] == string2[i] && string1[i] != '\0')
24.    {
25.        i++;
26.    }
27.
28.    if (string1[i] > string2[i])
29.        printf ("First string is greater than Second string\n");
30.    else if (string1[i] < string2[i])
31.        printf ("Second string is greater than First string\n");
32.    else
33.        printf ("Both strings are EQUAL\n");
34.
35.    return 0;
36.}
```

Program Explanation

1. Take two strings as input and store them in the arrays string1[] and string2[] respectively.
2. Count the number of characters in both the arrays and store the result in the variables count1 and count2.
3. Compare each character of the strings. If both the strings are equal then assign variable flag to zero or if string1 is greater than string2 then assign 1 to variable flag and break or if string1 is lesser than string2 then assign -1 to variable flag and break.
4. Print the output according to value of variable flag.

advertisement

Runtime Test Cases

```
Enter the First string
object
Enter the Second string
class
First string is greater than Second string
```

```
Enter the First string
object
Enter the Second string
object
Both strings are EQUAL
```

```
Enter the First string
class
Enter the Second string
object
Second string is greater than First string
```

30. C Program to Check if a String is a Palindrome without using the Built-in Function

[« Prev](#)

[Next »](#)

This is a C program to check a given string is palindrome without using the Built-in Function.

Problem Description

This program accepts a string and checks whether a given string is palindrome without using the built-in function.

Problem Solution

1. Take a string as input and store it in the array.
2. Reverse the string and store it in another array.
3. Compare both the arrays.

advertisement

Program/Source Code

Here is source code of the C program to check a given string is palindrome without using the Built-in Function .The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. /*
3.  * C program to find the length of a string without using the
4.  * built-in function also check whether it is a palindrome
5.  */
6. #include <stdio.h>
7. #include <string.h>
8.
9. void main()
10.{ 
11.     char string[25], reverse_string[25] = {"\0"};
12.     int i, length = 0, flag = 0;
13.
14.     printf("Enter a string \n");
15.     gets(string);
16.     /* keep going through each character of the string till its end */
17.     for (i = 0; string[i] != '\0'; i++)
18.     {
19.         length++;
20.     }
21.     printf("The length of the string '%s' = %d\n", string, length);
22.     for (i = length - 1; i >= 0 ; i--)
23.     {
24.         reverse_string[length - i - 1] = string[i];
25.     }
26.     /* Check if the string is a Palindrome */
27.
28.     for (flag = 1, i = 0; i < length ; i++)
29.     {
30.         if (reverse_string[i] != string[i])
31.             flag = 0;
32.     }
33.     if (flag == 1)
34.         printf ("%s is a palindrome \n", string);
35.     else
36.         printf("%s is not a palindrome \n", string);
```

Program Explanation

1. Take a string as input and store it in the array string[].
2. Store the same string into the another array reverse_string[] in the reverse fashion.
3. Using for loop compare the elements of both the arrays.
4. If all the elements of the array are same, then it is a palindrome. Otherwise it is not a palindrome.

advertisement

Runtime Test Cases

```
Enter a string
how are you
The length of the string 'how are you' = 12
how are you is not a palindrome
```

```
Enter a string
madam
The length of the string 'madam' = 5
madam is a palindrome
```

```
Enter a string
mam
The length of the string 'mam' = 3
mam is a palindrome
```

31. C Program to Create a File & Store Information

[« Prev](#)

[Next »](#)

This C Program creates a file & store information. We frequently use files for storing information which can be processed by our programs. In order to store information permanently and retrieve it we need to use files and this program demonstrate file creation and writing data in that.

Here is source code of the C program to create a file & store information. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to create a file called emp.rec and store information
3. * about a person, in terms of his name, age and salary.
4. */
5. #include <stdio.h>
6.
7. void main()
8. {
9.     FILE *fptr;
10.    char name[20];
11.    int age;
12.    float salary;
13.
14. /* open for writing */
15. fptr = fopen("emp.rec", "w");
16.
17. if (fptr == NULL)
18. {
19.     printf("File does not exists \n");
20.     return;
21. }
22. printf("Enter the name \n");
23. scanf("%s", name);
24. fprintf(fptr, "Name = %s\n", name);
25. printf("Enter the age\n");
26. scanf("%d", &age);
27. fprintf(fptr, "Age = %d\n", age);
28. printf("Enter the salary\n");
29. scanf("%f", &salary);
30. fprintf(fptr, "Salary = %.2f\n", salary);
31. fclose(fptr);
32. }
```

advertisement

```
$ cc pgm95.c
$ a.out
Enter the name
raj
Enter the age
40
Enter the salary
4000000
```

32. C Program to Find the Frequency of the Word ‘the’ in a given Sentence

[« Prev](#)

[Next »](#)

This is a C program to find the frequency of the word ‘the’ in a given sentence.

Problem Description

This program takes the sentence as input and finds the frequency of the word ‘the’ in a given sentence.

Problem Solution

1. Take any sentence as input.
2. Check for the word ‘the’ in the input sentence.
3. Use a variable to keep the count of number of ‘the’ in the sentence.

advertisement

Program/Source Code

Here is source code of the C program to find the frequency of the word ‘the’ in a given sentence. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to accept a string and find the number of times the word
3. * 'the' appears in that string
4. */
5. #include <stdio.h>
6.
7. void main()
8. {
9.     int count = 0, i, times = 0, t, h, e, space;
10.    char string[100];
11.
12.    puts("Enter a string:");
13.    gets(string);
14.    /* Traverse the string to count the number of characters */
15.    while (string[count] != '\0')
16.    {
17.        count++;
18.    }
19.    /* Finding the frequency of the word 'the' */
20.    for (i = 0; i <= count - 3; i++)
21.    {
22.        t =(string[i] == 't' || string[i] == 'T');
23.        h =(string[i + 1] == 'h' || string[i + 1] == 'H');
24.        e =(string[i + 2] == 'e' || string[i + 2] == 'E');
25.        space =(string[i + 3] == ' ' || string[i + 3] == '\0');
26.        if((t && h && e && space) == 1)
27.            times++;
28.    }
29.    printf("Frequency of the word 'the' is %d\n", times);
30.}
```

Program Explanation

1. Take any sentence as input and store it in the array string[].
2. If the input string has 't', 'h', 'e' and ' ' consecutively, then store that values in the variables t, h, e and space respectively.
3. Use the variable times to count the number of 'the' in the input sentence. Increment the variable times if and only if the variables t, h, e and space have values in it.
4. Print the variable times as output and exit.

advertisement

Runtime Test Cases

Enter a string:

The gandhi jayanthi is celeberated on october 2 is the day
that he has born.

Frequency of the word 'the' is 2

33. C Program to Compute the Value of X ^ N

« [Prev](#)

[Next »](#)

This is a C Program to compute the value of x^n .

Problem Description

This C Program computes the Value of X^N .

Problem Solution

The program uses power function defined in math library.

advertisement

Program/Source Code

Here is source code of the C program to computes the Value of X^N . The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to compute the value of X ^ N given X and N as inputs
 */
#include <stdio.h>
#include <math.h>

long int power(int x, int n);

void main()
{
    long int x, n, xpown;

    printf("Enter the values of X and N \n");
    scanf("%ld %ld", &x, &n);
    xpown = power(x, n);
    printf("X to the power N = %ld\n", xpown);
}

/* Recursive function to computer the X to power N */
long int power(int x, int n)
{
    if (n == 1)
        return(x);
    else if (n % 2 == 0)
        /* if n is even */
        return (pow(power(x, n/2), 2));
    else
        /* if n is odd */
        return (x * power(x, n - 1));
}
```

Program Explanation

In this C program, library function `pow()` defined in `<math.h>` header file is used to compute mathematical functions. We are reading two integer values using ‘`x`’ and ‘`n`’ variables respectively and passing it to `power()` function to compute X^N .

advertisement

The function power() uses recursion to compute the value.

In the power() function, if n equals 1, we return the value x to the calling function main(). If n is even, then we are using math library pow() function to

If condition statement is used to check the value of ‘n’ variable is equal to 1. If the condition is true, execute the statement. Otherwise, if the condition is false, execute the elseif conditional statement. Compute the modulus of n variable value by 2 and check the value is equal to zero, if the condition is true then it will execute the statement. Otherwise, if the condition is false execute the else statement.

advertisement

Runtime Test Cases

```
$ cc pgm55.c -lm
$ a.out
Enter the values of X and N
2 5
X to the power N = 32
```

34. C Program to Illustrate Reading of Data from a File

[« Prev](#)

[Next »](#)

This C Program illustrates reading of data from a file. The program opens a file which is present. Once the file opens successfully, it uses libc fgetc() library call to read the content.

Here is source code of the C program to illustrate reading of data from a file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to illustrate how a file stored on the disk is read
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. void main()
8. {
9.     FILE *fptr;
10.    char filename[15];
11.    char ch;
12.
13.    printf("Enter the filename to be opened \n");
14.    scanf("%s", filename);
15.    /* open the file for reading */
16.    fptr = fopen(filename, "r");
17.    if (fptr == NULL)
18.    {
19.        printf("Cannot open file \n");
20.        exit(0);
21.    }
22.    ch = fgetc(fptr);
23.    while (ch != EOF)
24.    {
25.        printf ("%c", ch);
26.        ch = fgetc(fptr);
27.    }
28.    fclose(fptr);
29.}
```

advertisement

```
$ cc pgm96.c
$ a.out
Enter the filename to be opened
pgm95.c
/*
 * C program to create a file called emp.rec and store information
 * about a person, in terms of his name, age and salary.
 */

#include <stdio.h>

void main()
{
    FILE *fptr;
    char name[20];
    int age;
    float salary;

    fptr = fopen ("emp.rec", "w"); /* open for writing*/
```

```
if (fptr == NULL)
{
    printf("File does not exists \n");
    return;
}
printf("Enter the name \n");
scanf("%s", name);
fprintf(fptr, "Name = %s\n", name);
printf("Enter the age \n");
scanf("%d", &age);
fprintf(fptr, "Age = %d\n", age);
printf("Enter the salary \n");
scanf("%f", &salary);
fprintf(fptr, "Salary = %.2f\n", salary);
fclose(fptr);
}
```

35. C Program to Find the Length of a String without using the Built-in Function

[« Prev](#)

[Next »](#)

This is a C program to find the length of a string without using the built-in function.

Problem Description

This program takes a string as input and finds its length without using the built-in function.

Problem Solution

1. Take a string as input and store it in the array.
2. Using for loop count the number of characters in the array and store the result in a variable.
3. Print the variable as output.

advertisement

Program/Source Code

Here is source code of the C program to find the length of a string without using the built-in function. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2. /*  
3.  * C program to find the length of a string without using the  
4.  * built-in function  
5.  */  
6. #include <stdio.h>  
7.  
8. void main()  
9. {  
10.    char string[50];  
11.    int i, length = 0;  
12.  
13.    printf("Enter a string \n");  
14.    gets(string);  
15.    /* keep going through each character of the string till its end */  
16.    for (i = 0; string[i] != '\0'; i++)  
17.    {  
18.        length++;  
19.    }  
20.    printf("The length of a string is the number of characters in it \n");  
21.    printf("So, the length of %s = %d\n", string, length);  
22.}
```

Program Explanation

1. Take a string as input and store it in the array string[].
2. Using for loop count the number of characters in the array string[]. Use the variable length to keep the count of the characters in the array.
3. Do step-2 till the end of input string.
4. Print the variable length as output.

advertisement

Runtime Test Cases

Enter a string

Sanfoundry

The length of a string is the number of characters in it

So, the length of Sanfoundry = 10

36. C Program to Find First N Fibonacci Numbers

« [Prev](#)

[Next](#) »

This C Program calculate the Fibonacci numbers in the series. The first two numbers in the Fibonacci sequence are 0 and 1 and each subsequent number is the sum of the previous two. The formula for this program is: $F_n = F_{n-1} + F_{n-2}$

Here is source code of the C program to calculate the Fibonacci Numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to generate and print first N FIBONACCI numbers
3. * in the series.
4. */
5. #include <stdio.h>
6.
7. void main()
8. {
9.     int fib1 = 0, fib2 = 1, fib3, num, count = 0;
10.
11.    printf("Enter the value of num \n");
12.    scanf("%d", &num);
13.    printf("First %d FIBONACCI numbers are ... \n", num);
14.    printf("%d\n", fib1);
15.    printf("%d\n", fib2);
16.    count = 2; /* fib1 and fib2 are already used */
17.    while (count < num)
18.    {
19.        fib3 = fib1 + fib2;
20.        count++;
21.        printf("%d\n", fib3);
22.        fib1 = fib2;
23.        fib2 = fib3;
24.    }
25. }
```

advertisement

```
$ cc pgm10.c
$ a.out
Enter the value of num
15
First 15 FIBONACCI numbers are ...
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
```

37. C Program to Multiply given Number by 4 using Bitwise Operators

[« Prev](#)

[Next »](#)

This is a C Program to multiply given number by 4 using bitwise operators.

Problem Description

This C Program multiplies given number by 4 using bitwise operators.

Problem Solution

The bitwise operators are or, and, xor, not, left shift, right shift. Program uses left shift operator for this.

advertisement

Program/Source Code

Here is source code of the C program to multiply given number by 4 using bitwise operators. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to multiply given number by 4 using bitwise operators
 */
#include <stdio.h>

void main()
{
    long number, tempnum;

    printf("Enter an integer \n");
    scanf("%ld", &number);
    tempnum = number;
    /* left shift by two bits */
    number = number << 2;
    printf("%ld x 4 = %ld\n", tempnum, number);
}
```

Program Explanation

In this C program, we are reading the integer using ‘number’ variable, and assign the value of ‘number’ variable to ‘tempnum’ variable. The bitwise operators are, or, and, xor, not, left shift, right shift. The program uses left shift operator to the value of ‘number’ variable by two bits.

advertisement

Runtime Test Cases

```
$ cc pgm62.c
$ a.out
Enter an integer
450
450 x 4 = 1800
```

38. C Program to Calculate the Sum of cos(x) Series

[« Prev](#)

[Next »](#)

This is a C Program to calculate the sum of cos(x) series.

Problem Description

This C Program calculates the sum of cos(x) series.

Problem Solution

Take input from the user and perform operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C program to C Program calculates the sum of cos(x) series. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the sum of cos(x) series
 */
#include <stdio.h>
#include <math.h>

void main()
{
    int n, x1, i, j;
    float x, sign, cosx, fact;

    printf("Enter the number of the terms in a series\n");
    scanf("%d", &n);
    printf("Enter the value of x(in degrees)\n");
    scanf("%f", &x);
    x1 = x;
        /* Degrees to radians */
    x = x * (3.142 / 180.0);
    cosx = 1;
    sign = -1;
    for (i = 2; i <= n; i = i + 2)
    {
        fact = 1;
        for (j = 1; j <= i; j++)
        {
            fact = fact * j;
        }
        cosx = cosx + (pow(x, i) / fact) * sign;
        sign = sign * (-1);
    }
    printf("Sum of the cosine series = %7.2f\n", cosx);
    printf("The value of cos(%d) using library function = %f\n", x1,
        cos(x));
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the number of the terms and the degree value of the series using ‘n’ and ‘x’ variables. To find the sum of cos(x) series, the following formula is used.

$$\text{Cos}(x) = \cos x + (\text{pow}(x, i) / \text{fact}) * \text{sign}$$

advertisement

Runtime Test Cases

```
$ cc pgm63.c -lm
$ a.out
Enter the number of the terms in a series
3
Enter the value of x(in degrees)
90
Sum of the cosine series = -0.23
The value of cos(90) using library function = -0.000204
```

39. C Program to Read a Grade & Display the Equivalent Description

[« Prev](#)

[Next »](#)

This C program reads a grade and displays its equivalent description.

Problem Description

This program takes a grade as input and displays its equivalent description.

Problem Solution

1. Take the grade as input.
2. Use switch statement to verify the grade.
3. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to read a grade & display the equivalent description. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to accept a grade and declare the equivalent description
3. * if code is S, then print SUPER
4. * if code is A, then print VERY GOOD
5. * if code is B, then print FAIR
6. * if code is Y, then print ABSENT
7. * if code is F, then print FAILS
8. */
9. #include <stdio.h>
10.#include <ctype.h>
11.#include <string.h>
12.
13.void main()
14.{
15.    char remark[15];
16.    char grade;
17.
18.    printf("Enter the grade \n");
19.    scanf("%c", &grade);
20.    /* lower case letter to upper case */
21.    grade = toupper(grade);
22.    switch(grade)
23.    {
24.        case 'S':
25.            strcpy(remark, " SUPER");
26.            break;
27.        case 'A':
28.            strcpy(remark, " VERY GOOD");
29.            break;
30.        case 'B':
31.            strcpy(remark, " FAIR");
32.            break;
33.        case 'Y':
```

```
34. strcpy(remark, " ABSENT");
35. break;
36. case 'F':
37. strcpy(remark, " FAILS");
38. break;
39. default :
40. strcpy(remark, "ERROR IN GRADE \n");
41. break;
42. }
43. printf("RESULT : %s\n", remark);
44. }
```

Program Explanation

1. Take the letter as input and store it in the variable grade.
2. Convert the input letter into its uppercase using function toupper().
3. Using switch statement, verify the input letter.
4. If the letter is S, then copy the string " SUPER" into the variable remark and break.
5. If the letter is A, then copy the string " VERY GOOD" into the variable remark and break.
6. If the letter is B, then copy the string " FAIR" into the variable remark and break.
7. If the letter is Y, then copy the string " ABSENT" into the variable remark and break.
8. If the letter is F , then copy the string " FAILS" into the variable remark and break.
9. In the default case, copy the string " ERROR IN GRADE" into the variable remark and break.
10. Print the variable remark as output and exit.

advertisement

Runtime Test Cases

Enter the grade
s
RESULT : SUPER

Enter the grade
a
RESULT : VERY GOOD

Enter the grade
b
RESULT : FAIR

Enter the grade
y
RESULT : ABSENT

Enter the grade
f
RESULT : FAILS

40. C Program to Accept the Height of a Person & Categorize as Taller, Dwarf & Average

[« Prev](#)

[Next »](#)

This C program accepts the height of a person & categorize as taller, dwarf & average.

Problem Description

This program accepts the height of a person as input and categorizes as taller, dwarf & average.

Problem Solution

1. Take the height of a person as input.
2. Using if,else statements, categorize it as taller, dwarf & average and print the output accordingly.
3. Exit.

advertisement

Program/Source Code

Here is source code of the C program to accept the height of a person & categorize as taller, dwarf & average. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to accept the height of a person in centimeter and
3. * categorize the person based on height as taller, dwarf and
4. * average height person
5. */
6.
7. #include <stdio.h>
8. void main()
9. {
10. float height;
11.
12. printf("Enter the Height (in centimetres) \n");
13. scanf("%f", &height);
14. if (height < 150.0)
15.     printf("Dwarf\n");
16. else if ((height >= 150.0) && (height <= 165.0))
17.     printf(" Average Height \n");
18. else if ((height > 165.0) && (height <= 195.0))
19.     printf("Taller \n");
20. else
21.     printf("Abnormal height \n");
22. }
```

Program Explanation

1. Take the height of a person as input and store it in the variable height.
2. If the variable height is lesser than 150 cm, then print the output as “Dwarf”.
3. If the variable height is lesser than or equal to 165 cm and greater than or equal to 150 cm, then print the output as “Average Height”.
4. If the variable height is lesser than or equal to 195 cm and greater than 165 cm, then print the output as “Taller”.
5. Exit.

Runtime Test Cases

Enter the Height (in centimetres)

165

Average Height

Enter the Height (in centimetres)

140

Dwarf

Enter the Height (in centimetres)

190

Taller

41. C Program to Simulate a Simple Calculator

[« Prev](#)

[Next »](#)

This is a C Program to simulate a simple calculator.

Problem Description

This C Program simulates a simple calculator.

Problem Solution

This program performs arithmetic operations like addition, subtraction, multiplication & division. Assume that the 2 numbers a & b are given. For the given element we need to perform addition, subtraction, multiplication & division.

advertisement

Program/Source Code

Here is source code of the C program which simulates a simple calculator. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to simulate a simple calculator to perform arithmetic
 * operations like addition, subtraction, multiplication and division
 */
#include <stdio.h>

void main()
{
    char operator;
    float num1, num2, result;

    printf("Simulation of a Simple Calculator\n");
    printf("*****\n");
    printf("Enter two numbers \n");
    scanf("%f %f", &num1, &num2);
    fflush(stdin);
    printf("Enter the operator [+,-,*,/] \n");
    scanf("%c", &operator);
    switch(operator)
    {
        case '+': result = num1 + num2;
                     break;
        case '-': result = num1 - num2;
                     break;
        case '*': result = num1 * num2;
                     break;
        case '/': result = num1 / num2;
                     break;
        default : printf("Error in operationn");
                     break;
    }
    printf("\n %5.2f %c %5.2f = %5.2f\n", num1, operator, num2, result);
}
```

Program Explanation

In this C program reading two integers and operator symbol using ‘num1’, ‘num2’ and ‘operator’ variables respectively.

advertisement

Switch case statement is used to perform arithmetic operations like addition, subtraction, multiplication and division in each case. If the operator symbol does not match in the switch case statement then execute the default statement. Print the statement as “Error in operation”.

Runtime Test Cases

```
$ cc pgm.c
$ a.out8
Simulation of a Simple Calculator
*****
Enter two numbers
2 3
Enter the operator [+,-,*,/]
+
2.00 + 3.00 = 5.00

$ a.out
Simulation of a Simple Calculator
*****
Enter two numbers
50 40
Enter the operator [+,-,*,/]
*
50.00 * 40.00 = 2000.00

$ a.out
Simulation of a Simple Calculator
*****
Enter two numbers
500 17
Enter the operator [+,-,*,/]
/
500.00 / 17.00 = 29.41

$ a.out
Simulation of a Simple Calculator
*****
Enter two numbers
65000 4700
Enter the operator [+,-,*,/]
-
65000.00 - 4700.00 = 60300.00
```

42. C Program to Read a String and find the Sum of all Digits in the String

[« Prev](#)

[Next »](#)

This is a C program to read a string and find the sum of all digits in the string.

Problem Description

This program takes a string containing both digits and alphabet as input and finds the sum of all digits in the string.

Problem Solution

1. Take the string as input.
2. Check for the digits in the string.
3. Count the number of digits and add all the digits to get the sum.

advertisement

Program/Source Code

Here is source code of the C program to read a string and find the sum of all digits in the string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2.  
3. /*  
4.  * C program to find the sum of all digits present in the string  
5. */  
6. #include <stdio.h>  
7. void main()  
8. {  
9.     char string[80];  
10.    int count, nc = 0, sum = 0;  
11.  
12.    printf("Enter the string containing both digits and alphabet \n");  
13.    scanf("%s", string);  
14.    for (count = 0; string[count] != '\0'; count++)  
15.    {  
16.        if ((string[count] >= '0') && (string[count] <= '9'))  
17.        {  
18.            nc += 1;  
19.            sum += (string[count] - '0');  
20.        }  
21.    }  
22.    printf("NO. of Digits in the string = %d\n", nc);  
23.    printf("Sum of all digits = %d\n", sum);  
24.}
```

Program Explanation

1. Take the string containing both digits and alphabet as input and store it in the array string[].
2. Using for loop and if statement check for the digits in the array. If it is, then increment the variable nc by 1 and increment the variable sum with the current digit.
3. Do the step-2 till the end of the input string.

4. Variable nc gives the count of number of digits in the array and variable sum gives the sum of all the digits in the array.

advertisement

Runtime Test Cases

Enter the string containing both digits and alphabet

hello100

NO. of Digits in the string = 3

Sum of all digits = 1

43. C Program to Compute the Sum of Digits in a given Integer

[« Prev](#)

[Next »](#)

This is a C program to compute the sum of digits in a given integer.

Problem Description

This program computes the sum of digits in a given integer.

Problem Solution

1. Take the integer as input.
2. Divide the input integer by 10, obtain its remainder and quotient.
3. Increment the new variable with the remainder got at step 2.
4. Repeat the step 2 & 3 with the quotient obtained until the quotient becomes zero.
5. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to compute the sum of digits in a given integer. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to accept an integer & find the sum of its digits
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     long num, temp, digit, sum = 0;
9.
10.    printf("Enter the number \n");
11.    scanf("%ld", &num);
12.    temp = num;
13.    while (num > 0)
14.    {
15.        digit = num % 10;
16.        sum = sum + digit;
17.        num /= 10;
18.    }
19.    printf("Given number = %ld\n", temp);
20.    printf("Sum of the digits %ld = %ld\n", temp, sum);
21. }
```

Program Explanation

1. Take an integer as a input and store it in the variable num.
2. Initialize the variable sum to zero.
3. Divide the input integer by 10 and obtain its remainder & quotient.
4. Store the remainder in the variable digit.
5. Increment the variable sum with variable digit.

6. Store the quotient into the variable num.
7. Repeat the steps 3,4,5,6 with the new num.
8. Do step 7 until the quotient becomes zero.
9. Print the variable sum as output and exit.

advertisement

Runtime Test Cases

Enter the number

300

Given number = 300

Sum of the digits 300 = 3

Enter the number

16789

Given number = 16789

Sum of the digits 16789 = 31

44. C Program to Replace Lowercase Characters by Uppercase & Vice-Versa

[« Prev](#)

[Next »](#)

This is a C program to replace lowercase characters by uppercase & vice-versa.

Problem Description

This program accepts the sentence and replaces lowercase characters by uppercase & vice-versa.

Problem Solution

1. Take the sentence as input.
2. Using (islower()): toupper():tolower() function replace lowercase characters by uppercase & vice-versa.
3. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to replace lowercase characters by uppercase & vice-versa. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to read an English sentence and replace
3. * lowercase characters by uppercase and vice-versa.
4. * Output the given sentence as well as the converted
5. * sentence on two different lines.
6. */
7. #include <stdio.h>
8. #include <ctype.h>
9.
10.void main()
11.{  
12.    char sentence[100];
13.    int count, ch, i;
14.
15.    printf("Enter a sentence \n");
16.    for (i = 0; (sentence[i] = getchar()) != '\n'; i++)
17.    {
18.        ;
19.    }
20.    sentence[i] = '\0';
21.    /* shows the number of chars accepted in a sentence */
22.    count = i;
23.    printf("The given sentence is : %s", sentence);
24.    printf("\n Case changed sentence is: ");
25.    for (i = 0; i < count; i++)
26.    {
27.        ch = islower(sentence[i])? toupper(sentence[i]):
28.            tolower(sentence[i]);
29.        putchar(ch);
30.    }
```

Program Explanation

1. Take an English sentence as input and store it in the array sentence[].
2. Copy the last letter's position in the array to the variable count.
3. Using for loop and (islower())? toupper():tolower() function replace lowercase characters by uppercase & vice-versa. Store this in the variable ch.
4. Print the variable ch as output and exit.

advertisement

Runtime Test Cases

```
Enter a sentence
wELCOME tO sANFOUNDRY
The given sentence is : wELCOME tO sANFOUNDRY
Case changed sentence is: Welcome To Sanfoundry
```

45. C Program to Count the Number of Vowels & Consonants in a Sentence

[« Prev](#)

[Next »](#)

This is a C program to count the number of vowels & consonants in a sentence.

Problem Description

This program takes the sentence as input and counts the number of vowels & consonants in a sentence.

Problem Solution

1. Take the sentence as input.
2. Using for loop and if,else statements check for vowels, consonants separately.
3. Print the output accordingly and exit.

advertisement

Program/Source Code

Here is source code of the C program to count the number of vowels & consonants in a sentence. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to read a sentence and count the total number of vowels
3. * and consonants in the sentence.
4. */
5. #include <stdio.h>
6.
7. void main()
8. {
9.     char sentence[80];
10.    int i, vowels = 0, consonants = 0, special = 0;
11.
12.    printf("Enter a sentence \n");
13.    gets(sentence);
14.    for (i = 0; sentence[i] != '\0'; i++)
15.    {
16.        if ((sentence[i] == 'a' || sentence[i] == 'e' || sentence[i] ==
17.             'i' || sentence[i] == 'o' || sentence[i] == 'u') ||
18.             (sentence[i] == 'A' || sentence[i] == 'E' || sentence[i] ==
19.             'I' || sentence[i] == 'O' || sentence[i] == 'U'))
20.        {
21.            vowels = vowels + 1;
22.        }
23.        else
24.        {
25.            consonants = consonants + 1;
26.        }
27.        if (sentence[i] == '\t' || sentence[i] == '\0' || sentence[i] == '\n')
28.        {
29.            special = special + 1;
30.        }
31.    }
32.    consonants = consonants - special;
33.    printf("No. of vowels in %s = %d\n", sentence, vowels);
```

```
34. printf("No. of consonants in %s = %d\n", sentence, consonants);  
35. }
```

Program Explanation

1. Take the sentence as input and store in the array sentence[].
2. Initialize the variables vowels, consonants and special to zero.
3. Using if,else statements, check if the sentence has vowels like a,e,i,o,u,A,E,I,O and U.
4. If it has, then increment the variable vowels by 1. Otherwise increment the variable consonants by 1.
5. If the sentence has \t, \0, & empty space, then increment the variable special by 1.
6. Do steps 3, 4 & 5 inside a for loop.
7. When for loop terminates, subtract the variable consonants from special.
8. Print the variables vowels and consonants as output.

advertisement

Runtime Test Cases

```
Enter a sentence  
welcome to sanfoundry  
No. of vowels in welcome to sanfoundry = 7  
No. of consonants in welcome to sanfoundry = 12
```

46. C Program to Find out the Roots of a Quadratic Equation

[« Prev](#)

[Next »](#)

This is a C Program to find out the roots of a quadratic equation.

Problem Description

This C Program calculates the roots of a quadratic equation.

Problem Solution

First it finds discriminant using the formula : $\text{disc} = b * b - 4 * a * c$. There are 3 types of roots. They are complex, distinct & equal roots. We have to find the given equation belongs to which type of root.

advertisement

Program/Source Code

Here is source code of the C program to calculate the roots of a quadratic equation. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find out the roots of a quadratic equation
 * for non-zero coefficients. In case of errors the program
 * should report suitable error message.
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void main()
{
    float a, b, c, root1, root2;
    float realp, imagp, disc;

    printf("Enter the values of a, b and c \n");
    scanf("%f %f %f", &a, &b, &c);
    /* If a = 0, it is not a quadratic equation */
    if (a == 0 || b == 0 || c == 0)
    {
        printf("Error: Roots cannot be determined \n");
        exit(1);
    }
    else
    {
        disc = b * b - 4.0 * a * c;
        if (disc < 0)
        {
            printf("Imaginary Roots\n");
            realp = -b / (2.0 * a);
            imagp = sqrt(abs(disc)) / (2.0 * a);
            printf("Root1 = %f +i %f\n", realp, imagp);
            printf("Root2 = %f -i %f\n", realp, imagp);
        }
        else if (disc == 0)
```

```

{
    printf("Roots are real and equal\n");
    root1 = -b / (2.0 * a);
    root2 = root1;
    printf("Root1 = %f\n", root1);
    printf("Root2 = %f\n", root2);
}
else if (disc > 0 )
{
    printf("Roots are real and distinct \n");
    root1 =(-b + sqrt(disc)) / (2.0 * a);
    root2 =(-b - sqrt(disc)) / (2.0 * a);
    printf("Root1 = %f \n", root1);
    printf("Root2 = %f \n", root2);
}
}
}

```

Program Explanation

In this C program, we are reading three integer values using ‘a’, ‘b’ and ‘c’ variables. If else condition statement is used to check the values are equal to 0. If the condition is true, then it is not a quadratic equation execute the statement and print as Error: Roots cannot be determined.

advertisement

Otherwise, if the condition is false, then execute the else statement. To find the discriminant value, the following formula is used

$$\text{Disc} = b * b - 4 * a * c.$$

Nested if else condition statement is used to display the 3 types of roots they are complex, distinct & equal roots from the equation.

advertisement

If the equation value in ‘disc’ variable is less than 0, then it is imaginary roots, execute the statement. To compute the real part and imaginary part the following formula is used

$$\begin{aligned}\text{Real part} &= -b / (2.0 * a) \\ \text{Imaginary part} &= \sqrt{|\text{disc}|} / (2.0 * a)\end{aligned}$$

Otherwise, if the condition is false, then execute the elseif condition statement. Check the value of ‘disc’ variable is equal to 0. If the condition is true, then roots are real and equal, execute the statement. To compute the real part and imaginary part the following formula is used

$$\begin{aligned}\text{Real part} &= -b / (2.0 * a) \\ \text{Real part} &= \text{Imaginary part}\end{aligned}$$

advertisement

If both the condition statements is false, then execute another else if statement. Check the value of ‘disc’ variable is greater than 0. If the condition is true, then the roots are real and distinct, execute the statement. To compute the real part and imaginary part the following formula is used

$$\begin{aligned}\text{Real part} &= (-b + \sqrt{\text{disc}}) / (2.0 * a) \\ \text{Imaginary part} &= (-b - \sqrt{\text{disc}}) / (2.0 * a)\end{aligned}$$

Runtime Test Cases

```
$ cc pgm7.c -lm
$ a.out
Enter the values of a, b and c
45 50 65
Imaginary Roots
Root1 = -0.555556 +i 1.065740
Root2 = -0.555556 -i 1.065740
```

47. C Program to Find the Sum of First N Natural Numbers

[« Prev](#)

[Next »](#)

This C Program calculates the sum of first N natural numbers.

Here is source code of the C program to calculate the sum of first N natural numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to find the sum of 'N' natural numbers
3.  */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int i, num, sum = 0;
9.
10.    printf("Enter an integer number \n");
11.    scanf ("%d", &num);
12.    for (i = 1; i <= num; i++)
13.    {
14.        sum = sum + i;
15.    }
16.    printf ("Sum of first %d natural numbers = %d\n", num, sum);
17.}
```

advertisement

```
$ cc pgm3.c
$ a.out
Enter an integer number
1300
Sum of first 1300 natural numbers = 845650
```

```
$ a.out
Enter an integer number
15
Sum of first 15 natural numbers = 120
```

48. C Program to Find the Largest Two Numbers in a given Array

[« Prev](#)

[Next »](#)

This is a C Program to calculate the largest two numbers in a given Array.

Problem Description

We have to write a program in C such that the program will read the elements of a one-dimensional array, then compares the elements and finds which are the largest two elements in a given array.

Expected Input and Output

1. Finding Largest 2 numbers in an array with unique elements:

advertisement

If we are entering 5 elements ($N = 5$), with array element values as 2,4,5,8 and 7 then,

The FIRST LARGEST = 8

THE SECOND LARGEST = 7

2. Finding Largest 2 numbers in an array with recurring elements:

If we are entering 6 elements ($N = 6$), with array element values as 2,1,1,2,1 and 2 then,

The FIRST LARGEST = 2

THE SECOND LARGEST = 1

Problem Solution

In this program, we have to find the largest and second-largest elements present in the array. We will do this by first saving the values of the first element in the variable ‘**largest**’ and second element in the variable ‘**second-largest**’. Then we will start from the third element and compare and swap with ‘**largest**’ and ‘**second-largest**’ numbers if another larger number is found in this array. This will go on $N-2$ times and the program ends.

advertisement

The sequence of steps for the solution will be as follows:

1. Create an array and define the elements of the array.
2. Considering the first element of the array to be the largest number and second element of the array to be the second largest element.
3. Interchange these two numbers if required.
4. Now run the loop from the third element of the array to the last element.
5. Scan element of the array, comparing array elements with the first largest and second-largest numbers, changing both or one if required.
6. In the end, after for loop, we will be getting the actual first largest and the second largest number in the array.

Program/Source Code

Here is the source code of the C program to calculate the largest of two numbers in a given array. The program is successfully compiled and tested using Turbo C compiler in the Windows environment. The program output is also shown below.

1. /*
2. * C program to read elements into an array and find the
3. * largest two elements in a given array.

```

4. */
5.
6. #include <stdio.h>
7. int main ()
8. {
9.     int n = 0, i = 0, largest1 = 0, largest2 = 0, temp = 0;
10.
11.    printf ("Enter the size of the array\n");
12.    scanf ("%d", &n);
13.    int array[n];
14.    printf ("Enter the elements\n");
15.    for (i = 0; i < n; i++)
16.    {
17.        scanf ("%d", &array[i]);
18.    }
19.
20.    printf ("The array elements are :\n");
21.    for (i = 0; i < n; i++)
22.    {
23.        printf ("%d\n", array[i]);
24.    }
25.
26.    printf ("\n");
27.
28.    largest1 = array[0];
29.    largest2 = array[1];
30.
31.    if (largest1 < largest2)
32.    {
33.        temp = largest1;
34.        largest1 = largest2;
35.        largest2 = temp;
36.    }
37.
38.    for (int i = 2; i < n; i++)
39.    {
40.        if (array[i] > largest1)
41.        {
42.            largest2 = largest1;
43.            largest1 = array[i];
44.        }
45.        else if (array[i] > largest2 && array[i] != largest1)
46.        {
47.            largest2 = array[i];
48.        }
49.    }
50.
51.    printf ("The FIRST LARGEST = %d\n", largest1);
52.    printf ("THE SECOND LARGEST = %d\n", largest2);
53.
54.    return 0;
55.}

```

Program Explanation

1. Declare an array of user-defined size.
2. Using for loop, define the elements of the array.
3. Consider the first element of array to be the first largest number (store it in a variable, **largest1**).
4. Consider the second element of array to be the second-largest number (store it in a variable, **largest2**).
5. Now interchange the values if the value of **largest1** is smaller than the **largest2**.
6. Run a for loop from the third element of the array till the last element of the array, wherein each element will be compared to the **largest1**.
 - i) If the value of the current element is larger than **largest1** then, we put the value of the current element to **largest1** and value of **largest1** to **largest2**.

ii) else if, the value of the current element is larger than the largest2 and is not equal to largest1, then we put the value of the current element to largest2.

advertisement

7. Running loop to the end will give us the actual **first largest and second-largest number**.

8. Exit

Runtime Test Cases

Here is the runtime output of the C program with 2 different test cases.

Test case 1: Here, the elements are unique. We are reading an array of 5 elements with unique values 2,4,5,8 and 7. The program is displaying the largest 2 numbers.

```
Enter the size of the array
```

```
5
```

```
Enter the elements
```

```
2
```

```
4
```

```
5
```

```
8
```

```
7
```

```
The array elements are :
```

```
2 4 5 8 7
```

```
The FIRST LARGEST = 8
```

```
THE SECOND LARGEST = 7
```

49. C Program to Convert the given Binary Number into Decimal

[« Prev](#)

[Next »](#)

This is a C program to convert binary number to decimal.

Problem Description

This program takes a binary number as input and converts it into decimal number.

Problem Solution

1. Take a binary number as input.
2. Multiply each digits of the binary number starting from the last with the powers of 2 respectively.
3. Add all the multiplied digits.
4. The total sum gives the decimal number.

advertisement

Program/Source Code

Here is source code of the C program to convert binary number to decimal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to convert the given binary number into decimal
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int num, binary_val, decimal_val = 0, base = 1, rem;
9.
10.    printf("Enter a binary number(1s and 0s) \n");
11.    scanf("%d", &num); /* maximum five digits */
12.    binary_val = num;
13.    while (num > 0)
14.    {
15.        rem = num % 10;
16.        decimal_val = decimal_val + rem * base;
17.        num = num / 10 ;
18.        base = base * 2;
19.    }
20.    printf("The Binary number is = %d \n", binary_val);
21.    printf("Its decimal equivalent is = %d \n", decimal_val);
22.}
```

Program Explanation

1. Take a binary number and store it in the variable num.
2. Initialize the variable decimal_val to zero and variable base to 1.
3. Obtain the remainder and quotient of the binary number. Store the remainder in the variable rem and override the variable num with quotient.
4. Multiply rem with variable base. Increment the variable decimal_val with this new value.
5. Increment the variable base by 2.

6. Repeat the steps 3, 4 and 5 with the quotient obtained until quotient becomes zero.

7. Print the variable decimal_val as output.

advertisement

Runtime Test Cases

Enter a binary number(1s and 0s)

10101001

The Binary number is = 10101001

Its decimal equivalent is = 169

50. C Program to Insert an Element in a Specified Position in a given Array

[« Prev](#)

[Next »](#)

This is a C Program to insert an element in a specified position in a given array.

Problem Description

This program implements a one dimensional array, sorts it and then takes a user input and inserts the desired element in the specified position of a one dimensional array and print all the elements of the array with a +1 increment in the array size.

Problem Solution

1. Declare a one dimensional array of some fixed capacity.
2. Take size of the array as input from users, which must be at least one less than array's capacity.
3. Define array elements, taking each element as input from users.
4. Sort the array elements.
5. Now, take a value from users, which needs to be inserted inside this array.
6. Select a suitable position for the new element by comparing the new element to the array elements and insert it in that position.
7. Print all the elements. Now, the array size would be previous array size+1.
8. Exit

advertisement

Program/Source Code

Here is source code of the C program to insert an element in a specified position in a given array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.
2. /*
3.  * C program to insert a particular element in a specified position
4.  * in a given array
5. */
6.
7. #include <stdio.h>
8. void main()
9. {
10.    int array[10];
11.    int i, j, n, m, temp, key, pos;
12.
13.    printf("Enter how many elements \n");
14.    scanf("%d", &n);
15.    printf("Enter the elements \n");
16.
17.    for (i = 0; i < n; i++)
18.    {
19.        scanf("%d", &array[i]);
20.    }
```

```

21.
22. printf("Input array elements are \n");
23. for (i = 0; i < n; i++)
24. {
25.     printf("%d\n", array[i]);
26. }
27.
28. // Sorting the elements of the array
29. for (i = 0; i < n; i++)
30. {
31.     for (j = i + 1; j < n; j++)
32.     {
33.         if (array[i] > array[j])
34.         {
35.             temp = array[i];
36.             array[i] = array[j];
37.             array[j] = temp;
38.         }
39.     }
40. }
41.
42. printf("Sorted list is \n");
43. for (i = 0; i < n; i++)
44. {
45.     printf("%d\n", array[i]);
46. }
47.
48. printf("Enter the element to be inserted \n");
49. scanf("%d", &key);
50.
51. for (i = 0; i < n; i++)
52. {
53.     if (key < array[i])
54.     {
55.         pos = i;
56.         break;
57.     }
58.     if (key > array[n-1])
59.     {
60.         pos = n;
61.         break;
62.     }
63. }
64. if (pos != n)
65. {
66.     m = n - pos + 1 ;
67.     for (i = 0; i <= m; i++)
68.     {
69.         array[n - i + 2] = array[n - i + 1] ;
70.     }
71. }
72.
73. array[pos] = key;
74.
75. printf("Final list is \n");
76. for (i = 0; i < n + 1; i++)
77. {
78.     printf("%d\n", array[i]);
79. }
80.
81. }
```

Program Explanation

1. Declare a one dimensional array, a of some fixed capacity, 10.
2. Take an input for size of the array, n. The size of the array must be at least one smaller than array's capacity,

- 10.
3. Using for loop, fill the array with elements to its size.
4. Print all the elements of the array.
5. Sort the elements of the array in ascending order. In this case, we have used Insertion sort.
6. Now, take a value from users, which needs to be inserted inside array.
7. Run a for loop from 0 to arraySize-1, checking and comparing each element of array
- i) If new element is smaller than array element, then the position is array element is saved.
- ii) If new element is bigger than the last element of array (biggest element of array), then the size of array is saved.
8. Position saved is where the new element will be inserted.
9. All the elements at that position in the array and after that are shifted backwards by 1.
10. Print all the elements of the new array.

advertisement

Runtime Test Cases

Enter how many elements

5

Enter the elements

76

90

56

78

12

Input array elements are

76

90

56

78

12

Sorted list is

12

56

76

78

90

Enter the element to be inserted

61

Final list is

12

56

61

76

78

90

51. C Program to Calculate the Sum of the Array Elements using Pointer

[« Prev](#)

[Next »](#)

This is a C program to calculate sum of array elements using pointer.

Problem Description

We have to write a program in C such that it calculates the sum of elements of an array using pointers. The program should dynamically allocate a piece of memory for that array and use a pointer to point to that array memory as well as traverse that array using a pointer.

Expected Input and Output

If we are entering 5 elements ($N = 5$), with array element values as 4, 9, 10, 56 and 100 then,

Sum of Elements of the array will be: $4 + 9 + 10 + 56 + 100 = 179$
advertisement

Problem Solution

Pointers are used to store Address of variables or a memory location. In C, a pointer is created by placing a * before the variable name. We can access the value of the pointer by using * before the name of the pointer variable.

For example, `int *x; /* this creates a pointer with name x */`

This program uses `malloc()` function of C. This function is used for memory allocation. The syntax of malloc is
advertisement

pointer_name = (cast-type*) malloc (byte-size)

For example, if you have to allocate 20 bytes of storage to an integer pointer named `ptr`, then we will use the following C code:

`int *ptr = (int*) malloc (5 * sizeof(int));`

advertisement

The sequence of steps for the solution will be as follows:

1. Create a pointer variable, which points to an integer data.
2. Take a number i.e size of array as input.
3. Create a block of space fo size (`size-of-array*sizeof(int)`) using `malloc()` assigning the starting address of this whole space to the pointer variable.
4. Iterate via `for` loop for reading array elements as input.
5. Iterate again via `for` loop to access the address stored in pointer variable, adding the iterator value to it, so as to access every element of array, and calculating the overall sum.

Program/Source Code

Here is source code of the C program to calculates the sum of array elements using pointer. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

1. /*
2. * C program to read N integers and store them in an array A.
3. * Find the sum of all these elements using pointer.

```

4.  */
5.
6. #include <stdio.h>
7. #include <malloc.h>
8.
9. void main()
10. {
11.     int i, n, sum = 0;
12.     int *a;
13.
14.     printf("Enter the size of array A \n");
15.     scanf("%d", &n);
16.
17.     a = (int *) malloc(n * sizeof(int));
18.
19.     printf("Enter Elements of the List \n");
20.     for (i = 0, i < n; i++)
21.     {
22.         scanf("%d", a + i);
23.     }
24.
25.     /* Compute the sum of all elements in the given array */
26.
27.     for (i = 0; i < n; i++)
28.     {
29.         sum = sum + *(a + i);
30.         /* this *(a+i) is used to access the value stored at the address*/
31.     }
32.
33.     printf("Sum of all elements in array = %d\n", sum);
34.     return 0;
35. }
```

Program Explanation

1. Create a pointer variable **a**, which points to an integer data.
2. Now, create another variable **n** (size of array), whose input should be taken by users, and a variable **sum** (initial value of sum = 0), which will store the total sum of all the elements of the array.
3. Now allocate **size_of_array(N) * size_of_each_element (integer)** times space using **malloc()** function, assigning the starting address of this space to the pointer variable, **a**.
4. Using **for** loop (0 to size), start iteration, reach at every array element location by adding the value of **i** to the value of **a**, taking array element as input.
5. Again, start iteration, this time reach each array element location, and accessing them to add to the **sum** variable.
6. Print **sum**.

advertisement

Runtime Test Cases

Here is the runtime output of the C program where the user is reading an array of 5 integers with values 4, 9, 10, 56 and 100 and the program is calculating the sum of the elements of the array using pointer and then displaying the result.

```

Enter the size of array A
5
Enter Elements of the List
4
9
10
56
100
Sum of all elements in array = 179
```

52. C Program to Check if a given String is Palindrome

[« Prev](#)

[Next »](#)

This is a C program to check a given string is palindrome.

Problem Description

This program accepts a string and checks whether a given string is palindrome.

Problem Solution

1. Take a string as input and store it in the array.
2. Reverse the string and store it in another array.
3. Compare both the arrays.

advertisement

Program/Source Code

Here is source code of the C program to check a given string is palindrome. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. /*
3. * C program to read a string and check if it's a palindrome, without
4. * using library functions. Display the result.
5. */
6. #include <stdio.h>
7. #include <string.h>
8.
9. void main()
10. {
11.     char string[25], reverse_string[25] = {"\0"};
12.     int i, length = 0, flag = 0;
13.
14.     fflush(stdin);
15.     printf("Enter a string \n");
16.     gets(string);
17.     /* keep going through each character of the string till its end */
18.     for (i = 0; string[i] != '\0'; i++)
19.     {
20.         length++;
21.     }
22.     for (i = length - 1; i >= 0; i--)
23.     {
24.         reverse_string[length - i - 1] = string[i];
25.     }
26.     /*
27.     * Compare the input string and its reverse. If both are equal
28.     * then the input string is palindrome.
29.     */
30.     for (i = 0; i < length; i++)
31.     {
32.         if (reverse_string[i] == string[i])
33.             flag = 1;
```

```
34.     else
35.         flag = 0;
36.     }
37.     if (flag == 1)
38.         printf("%s is a palindrome \n", string);
39.     else
40.         printf("%s is not a palindrome \n", string);
41. }
```

Program Explanation

1. Take a string as input and store it in the array string[].
2. Store the same string into the another array reverse_string[] in the reverse fashion.
3. Using for loop compare the elements of both the arrays.
4. If all the elements of the array are same, then it is a palindrome. Otherwise it is not a palindrome.

advertisement

Runtime Test Cases

```
Enter a string
sanfoundry
sanfoundry is not a palindrome
```

```
Enter a string
malayalam
malayalam is a palindrome
```

53. C Program to Read Two Integers M and N & Swap their Values

[« Prev](#)

[Next »](#)

This is a C program to read two integers & swap their values.

Problem Description

This program reads two integers & swaps their values.

Problem Solution

1. Take two integers as input.
2. Use a function for swapping process.
3. Exchange the values using another variable.
4. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C program to read two integers & swap their values. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to read two integers M and N and to swap their values.
3.  * Use a user-defined function for swapping. Output the values of M
4.  * and N before and after swapping.
5. */
6. #include <stdio.h>
7. void swap(float *ptr1, float *ptr2);
8.
9. void main()
10. {
11.     float m, n;
12.
13.     printf("Enter the values of M and N \n");
14.     scanf("%f %f", &m, &n);
15.     printf("Before Swapping:M = %5.2fN = %5.2f\n", m, n);
16.     swap(&m, &n);
17.     printf("After Swapping:M = %5.2fN = %5.2f\n", m, n);
18. }
19./* Function swap - to interchanges the contents of two items */
20.void swap(float *ptr1, float *ptr2)
21.{
22.    float temp;
23.
24.    temp = *ptr1;
25.    *ptr1 = *ptr2;
26.    *ptr2 = temp;
27.}
```

Program Explanation

1. Take the two integers as input and store it in the variables m and n respectively.
2. Call function swap. Pass addresses of variables to the function swap.
3. Receive the addresses by the two pointers ptr1 and ptr2.
4. First copy the value stored at &m to the variable temp.
5. Copy the value stored at &n to &m.
6. Copy the value in the variable temp to &n.
7. Print the variables m and n as output and exit

advertisement

Runtime Test Cases

Enter the values of M and N

2 3

Before Swapping:M = 2.00 N = 3.00

After Swapping:M = 3.00 N = 2.00

54. C Program to Check if a given Number is Prime number

[« Prev](#)

[Next »](#)

This is a C program to check whether a given number is prime or not.

Problem Description

This program takes a number and checks whether a given number is prime or not.

Problem Solution

1. Take a number as input.
2. Check if the number is divisible by any of the natural numbers starting from 2.
3. If it is, then it is not a prime number. Otherwise it is a prime number.
4. Exit.

advertisement

Program/Source Code

Here is source code of the C program to check if a given number is prime. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to check whether a given number is prime or not
3. * and output the given number with suitable message.
4. */
5. #include <stdio.h>
6. #include <stdlib.h>
7.
8. void main()
9. {
10.    int num, j, flag;
11.
12.    printf("Enter a number \n");
13.    scanf("%d", &num);
14.
15.    if (num <= 1)
16.    {
17.        printf("%d is not a prime numbers \n", num);
18.        exit(1);
19.    }
20.    flag = 0;
21.    for (j = 2; j <= num / 2; j++)
22.    {
23.        if ((num % j) == 0)
24.        {
25.            flag = 1;
26.            break;
27.        }
28.    }
29.    if (flag == 0)
```

```
30.     printf("%d is a prime number \n", num);
31. else
32.     printf("%d is not a prime number \n", num);
33. }
```

Program Explanation

1. Take a number as input and store it in the variable num.
2. If the number is lesser than or equal to 1, then print the output as " It is not a prime number".
3. Initialize the variable flag to zero.
4. Using for loop, check if the input number is divisible by any of the natural numbers starting from 2.
5. If it is, then assign the variable flag with 1.
6. Print the output as "It is a prime number", if the variable flag ==0.
7. Otherwise print the output as "It is not a prime number" and exit.

advertisement

Runtime Test Cases

```
Enter a number
23
23 is a prime number
```

```
Enter a number
15
15 is not a prime number
```

55. C Program to Find the GCD and LCM of Two Integers

[« Prev](#)

[Next »](#)

This C Program calculates the GCD and LCM of two integers. Here GCD means Greatest Common Divisor. For two integers a and b , if there are any numbers d so that a / d and b / d doesn't have any remainder, such a number is called a common divisor. Common divisors exist for any pair of integers a and b , since we know that 1 always divides any integer. We also know that common divisors can't get too big since divisors can't be any larger than the number they are dividing. Hence a common divisor d of a and b must have $d \leq a$ and $d \leq b$. Here, LCM means Least Common Multiplies. For two integer a & b , to know if there are any smallest numbers d so that d / a and d / b doesn't have a remainder. such a number is called a Least Common Multiplier.

Here is source code of the C program to calculate the GCD and LCM of two integers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to find the GCD and LCM of two integers using Euclid's algorithm
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int num1, num2, gcd, lcm, remainder, numerator, denominator;
9.
10.    printf("Enter two numbers\n");
11.    scanf("%d %d", &num1, &num2);
12.    if (num1 > num2)
13.    {
14.        numerator = num1;
15.        denominator = num2;
16.    }
17.    else
18.    {
19.        numerator = num2;
20.        denominator = num1;
21.    }
22.    remainder = numerator % denominator;
23.    while (remainder != 0)
24.    {
25.        numerator = denominator;
26.        denominator = remainder;
27.        remainder = numerator % denominator;
28.    }
29.    gcd = denominator;
30.    lcm = num1 * num2 / gcd;
31.    printf("GCD of %d and %d = %d\n", num1, num2, gcd);
32.    printf("LCM of %d and %d = %d\n", num1, num2, lcm);
33.}
```

advertisement

```
$ cc pgm11.c
$ a.out
Enter two numbers
30
```

40
GCD of 30 and 40 = 10
LCM of 30 and 40 = 120

56. C Program to Compute the Surface Area & Volume of a Cube

[« Prev](#)

[Next »](#)

This is a C Program to compute the surface area & volume of a cube.

Problem Description

This C Program computes the surface area & volume of a cube.

Problem Solution

The formula used to find the surface area and volume of the cube is surface_area = $6 * (a * a)$ and volume = $a * a * a$.

advertisement

Program/Source Code

Here is source code of the C program to compute the surface area & volume of a cube. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to compute the surface area and volume of a cube
 */
#include <stdio.h>
#include <math.h>

void main()
{
    float side, surfacearea, volume;

    printf("Enter the length of a side \n");
    scanf("%f", &side);
    surfacearea = 6.0 * side * side;
    volume = pow(side, 3);
    printf("Surface area = %6.2f and Volume = %6.2f \n", surfacearea,
          volume);
}
```

Program Explanation

In this C program, library function is used in header file to compute mathematical functions. We are entering the length of a side using side variable. Now to find the surface area of a cube the formula, surface area = $6 * (\text{side} * \text{side})$ is used. Then, to find the volume of a cube the formula, volume = $\text{pow}(\text{side}, 3)$ is used. Here, the program uses power function defined in math library. Finally, the surface area and volume will be displayed in the standard output.

advertisement

Runtime Test Cases

```
$ cc pgm45.c -lm
$ a.out
Enter the length of a side
34
Surface area = 6936.00 and Volume = 39304.00
```

57. C Program to Illustrate how User Authentication is Done

[« Prev](#)

[Next »](#)

This is a C program to illustrate user authentication.

Problem Description

This C program asks for the user name & password and displays the same to illustrate user authentication.

Problem Solution

1. Take the user name & password as input.
2. Print the each character of password as * while receiving it.
3. Now print the original password and exit.

advertisement

Program/Source Code

Here is source code of the C program to illustrate user authentication. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program is to illustrate how user authentication is done.
3.  * Program asks for the user name and password and displays
4.  * the password as '*' character
5. */
6. #include <stdio.h>
7.
8. void main()
9. {
10. char password[10], username[10], ch;
11. int i;
12.
13. printf("Enter User name: ");
14. gets(username);
15. printf("Enter the password < any 8 characters>: ");
16. for (i = 0; i < 8; i++)
17. {
18.     ch = getchar();
19.     password[i] = ch;
20.     ch = '*';
21.     printf("%c", ch);
22. }
23. password[i] = '\0';
24. /* Original password can be printed, if needed */
25. printf("\n Your password is :");
26. for (i = 0; i < 8; i++)
27. {
28.     printf("%c", password[i]);
```

```
29. }  
30. }
```

Program Explanation

1. Take the username as input and store it in the array username[].
2. Using for loop take the each character of password as input and store it in the array password[] and consecutively print it as '*'.
3. Print the array password[] as output and exit.

advertisement

Runtime Test Cases

```
Enter User name: rajaraman  
Enter the password <any 8 characters>: shashi12  
*****  
Your password is :shashi12
```

58. C Program to Check if the Substring is present in the given String

[« Prev](#)

[Next »](#)

This is a C program to check if the substring is present in the given string.

Problem Description

This C Program checks the substring is present in the given string.

Problem Solution

The program accepts both string and substring. Then checks whether the substring is present in the given string.

advertisement

Program/Source Code

Here is source code of the C program to check the substring is present in the given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to accept a string and a substring and
 * check if the substring is present in the given string
 */
#include<stdio.h>

int main()
{
    char str[80], search[10];
    int count1 = 0, count2 = 0, i, j, flag;

    printf("Enter a string:");
    gets(str);
    printf("Enter search substring:");
    gets(search);
    while (str[count1] != '\0')
        count1++;
    while (search[count2] != '\0')
        count2++;
    for (i = 0; i <= count1 - count2; i++)
    {
        for (j = i; j < i + count2; j++)
        {
            flag = 1;
            if (str[j] != search[j - i])
            {
                flag = 0;
                break;
            }
        }
        if (flag == 1)
            printf("Substring found at index %d", i);
        else
            printf("Substring not found");
    }
}
```

```

    }
}

if(flag == 1)
    break;
}
if(flag == 1)
    printf("SEARCH SUCCESSFUL!");
else
    printf("SEARCH UNSUCCESSFUL!");

return 0;
}

```

Program Explanation

In this C program, we are reading a string using `gets()` function ‘str’ character variable. We are reading value another string to search using search character variable. To check substring is present in the given string. While loop is used to compute the `str[]` and `search[]` array variable value is not equal to null.

advertisement

If the condition is true then execute the iteration of the loop. Increment the values of ‘count1’ and ‘count2’ variable values. Now we are using two for loops to check if the substring is present in the given string. We are initializing the ‘i’ variable value to 0 and the loop will execute till the condition that ‘i’ variable value should be less than or equal to the difference of `count1` and `count2` variable values.

If the condition is true, then another for loop will execute by initializing the ‘i’ variable value to ‘j’ variable. And the loop will terminate if the ‘j’ variable value is less than the sum of ‘i’ variable value with `count2` variable value if the condition is true.

Then it will execute the loop by assigning the `flag` variable value as 1 and if condition statement is used to check the `str[]` array variable value is not equal to `search[]` with the base index is the difference between ‘j’ variable and ‘i’ variable value. If the condition is true then it will execute the statement and assign `flag` variable value as 0. For loop iteration will terminate till the condition becomes false. If-else condition statement is used to check if `flag` variable value is equal to 1 then print as search successful otherwise if the condition is false then print the statement as search unsuccessful.

advertisement

Runtime Test Cases

```

$ cc pgm44.c
$ a.out
Enter a string: hello
Enter search substring: world
SEARCH UNSUCCESSFUL!

```

```

$ a.out
Enter a string: helloworld
Enter search substring:ld
SEARCH SUCCESSFUL!

```

59. C Program to Find the Sum of first 50 Natural Numbers using For Loop

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of first 50 natural numbers using for loop.

Problem Description

This C Program finds sum of first 50 natural numbers using for loop.

Problem Solution

It displays the sum of first 50 natural numbers as output.

advertisement

Program/Source Code

Here is source code of the C program to find the sum of first 50 natural numbers using for loop. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the sum of first 50 natural numbers
 * using for loop
 */
#include <stdio.h>

void main()
{
    int num, sum = 0;

    for (num = 1; num <= 50; num++)
    {
        sum = sum + num;
    }
    printf("Sum = %4d\n", sum);
}
```

Program Explanation

In this C program, we are reading the sum of first 50 natural numbers using for loop. By initializing the value of ‘num’ variable as 1 and checks the condition that ‘num’ variable value is less than or equal to 50. If the condition is true then we are computing the summation of the value of ‘sum’ variable with the value of ‘num’ variable. After that, we are displaying the sum of first 50 natural numbers as output.

advertisement

Runtime Test Cases

```
$ cc pgm73.c
$ a.out
Sum = 1275
```

60. C Program to Accept two Integers and Check if they are Equal

[« Prev](#)

[Next »](#)

This is a C program to accept two integers and check if they are equal.

Problem Description

This program accepts two integers and check if they are equal or not.

Problem Solution

1. Take the two integers as input.
2. Using if,else statements check if they are equal or not.
3. Print the output accordingly and exit.

advertisement

Program/Source Code

Here is source code of the C program to accepts two integers and check if they are equal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to accept two integers and check if they are equal
3. */
4. #include <stdio.h>
5. void main()
6. {
7.     int m, n;
8.
9.     printf("Enter the values for M and N\n");
10.    scanf("%d %d", &m, &n);
11.    if (m == n)
12.        printf("M and N are equal\n");
13.    else
14.        printf("M and N are not equal\n");
15. }
```

Program Explanation

1. Take the two integers as input and store it in the variables m and n respectively.
2. Using if,else statements check if m is equal to n.
3. If they are equal, then print the output as “M and N are equal”.
4. Otherwise print it as “M and N are not equal”.

advertisement

Runtime Test Cases

Case:1

Enter the values for M and N

3 3

M and N are equal

Case:2

Enter the values for M and N

61. C Program to Find the Areas of Different Geometrical Figures

[« Prev](#)

[Next »](#)

This is a C Program to find the areas of different geometrical figures.

Problem Description

This C Program finds the areas of different geometrical figures.

Problem Solution

The program is menu driven program. This has 4 options that can be chosen by the user. The 4 options are 1) area of circle 2) area of rectangle 3) area of triangle 4) area of square. All the 4 finds area.

advertisement

Program/Source Code

Here is source code of the C program to find the area of different geometrical figures. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to find the areas of different geometrical shapes such as
 * circle, square, rectangle etc using switch statements.
 */
#include <stdio.h>

void main()
{
    int fig_code;
    float side, base, length, breadth, height, area, radius;

    printf("-----\n");
    printf(" 1 --> Circle\n");
    printf(" 2 --> Rectangle\n");
    printf(" 3 --> Triangle\n");
    printf(" 4 --> Square\n");
    printf("-----\n");
    printf("Enter the Figure code\n");
    scanf("%d", &fig_code);
    switch(fig_code)
    {
        case 1:
            printf("Enter the radius\n");
            scanf("%f", &radius);
            area = 3.142 * radius * radius;
            printf("Area of a circle = %f\n", area);
            break;
        case 2:
            printf("Enter the breadth and length\n");
            scanf("%f %f", &breadth, &length);
            area = breadth * length;
            printf("Area of a Reactangle = %f\n", area);
            break;
    }
}
```

```

case 3:
    printf("Enter the base and height\n");
    scanf("%f %f", &base, &height);
    area = 0.5 * base * height;
    printf("Area of a Triangle = %f\n", area);
    break;

case 4:
    printf("Enter the side\n");
    scanf("%f", &side);
    area = side * side;
    printf("Area of a Square=%f\n", area);
    break;

default:
    printf("Error in figure code\n");
    break;
}
}

```

Program Explanation

In this C program, we are finding the areas of different geometrical figures. First we are displaying the options that can be chosen by the user. Then using switch case statement we are finding the area of the geometrical figures.

advertisement

In case1 we are reading the radius of a circle, calculating the area and displaying the result. The following formula is used to calculate the area of a circle

$$\text{Area} = 3.142 * \text{radius} * \text{radius}$$

In case2 we are reading the ‘breadth’ and ‘length’ of a rectangle, calculating the area and displaying the result. The following formula is used to calculate the area of a rectangle

$$\text{Area} = \text{breadth} * \text{length}$$

In case3 we are reading ‘base’ and ‘height’ of a triangle, calculating the area and displaying the result. The following formula is used to calculate the area of a triangle

$$\text{Area} = 0.5 * \text{base} * \text{height}$$

advertisement

In case4 we are reading the side of a square, calculating the area and displaying the result. The following formula is used to calculate the area of a circle

$$\text{Area} = \text{side} * \text{side}$$

If the user entered the number not in the menu strip then in default statement then display the output as an error in figure code.

Runtime Test Cases

```
$ cc pgm77.c
$ a.out
```

```
1 --> Circle
2 --> Rectangle
3 --> Triangle
4 --> Square
```

```
Enter the Figure code
1
Enter the radius
```

30

Area of a circle = 2827.800049

\$ a.out

1 --> Circle
2 --> Rectangle
3 --> Triangle
4 --> Square

Enter the Figure code

2

Enter the breadth and length

20 30

Area of a Reactangle = 600.000000

\$ a.out

1 --> Circle
2 --> Rectangle
3 --> Triangle
4 --> Square

Enter the Figure code

3

Enter the base and height

45 80

Area of a Triangle = 1800.000000

\$ a.out

1 --> Circle
2 --> Rectangle
3 --> Triangle
4 --> Square

Enter the Figure code

4

Enter the side

100

Area of a Square=10000.000000

62. C Program to Read a Coordinate Point in a XY Coordinate System and Determine its Quadrant

[« Prev](#)

[Next »](#)

This is a C Program to read a coordinate point in a xy coordinate system and determine its quadrant.

Problem Description

This C Program read a coordinate point in a XY coordinate system and determine its quadrant.

Problem Solution

The program accepts X and Y. Depending on the value of X and Y we need to determine on which quadrant this point lies.

advertisement

Program/Source Code

Here is source code of the C program to read a coordinate point in a XY coordinate system and determine its quadrant. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to accept a coordinate point in a XY coordinate system
 * and determine its quadrant
 */
#include <stdio.h>

void main()
{
    int x, y;

    printf("Enter the values for X and Y\n");
    scanf("%d %d", &x, &y);
    if (x > 0 && y > 0)
        printf("point (%d, %d) lies in the First quadrant\n");
    else if (x < 0 && y > 0)
        printf("point (%d, %d) lies in the Second quadrant\n");
    else if (x < 0 && y < 0)
        printf("point (%d, %d) lies in the Third quadrant\n");
    else if (x > 0 && y < 0)
        printf("point (%d, %d) lies in the Fourth quadrant\n");
    else if (x == 0 && y == 0)
        printf("point (%d, %d) lies at the origin\n");
}
```

Program Explanation

In this C program, we are determining the type of quadrant in XY quadrant system. We are reading the values for 'X' and 'Y' variable. Nested-if else condition system is used to determine the quadrant of the given value. If conditional statement is used to check the condition that 'X' variable value is greater than 0 and the 'Y' variable

value is greater than 0 using logical AND operator. If the condition is true then it will display the output as the first quadrant.

advertisement

Otherwise, if the condition is false then it will execute the else if conditional statement to check the condition that 'X' variable value is less than 0 and the 'Y' variable value is greater than 0 using logical AND operator. If the condition is true then it will display the output as the second quadrant.

If the condition is false then it will execute another elseif conditional statement to check the condition that 'X' variable value is less than 0 and the 'Y' variable value is less than 0 using logical AND operator. If the condition is true then it will display the output as the third quadrant.

Otherwise, if the condition is false then it will execute next elseif conditional statement to check the condition that 'X' variable value is greater than 0 and the Y variable value is less than 0 using logical AND operator. If the condition is true then it will display the output as the fourth quadrant.

advertisement

If the condition is false then it will execute next elseIf statement that the x variable value is equal to 0 and the Y variable value is equal to 0 using logical AND operator, then it will display the output as an origin.

Runtime Test Cases

```
$ cc pgm76.c
$ a.out
Enter the values for X and Y
20 30
point (-1079549476, -1079549480) lies in the First quadrant

$ a.out
Enter the values for X and Y
-30 -60
point (-1080802740, -1080802744) lies in the Third quadrant

$ a.out
Enter the values for X and Y
300 -8
point (-1078902004, -1078902008) lies in the Fourth quadrant

$ a.out
Enter the values for X and Y
-180 180
point (-1076456724, -1076456728) lies in the Second quadrant
```

63. C Program to Find the Size of a Union

[« Prev](#)

[Next »](#)

This is a C program to find the size of a Union.

Problem Description

This Program finds the size of a Union.

Problem Solution

1. Define the union.
2. Finds its size using keyword sizeof().
3. Print the same and exit.

advertisement

Program/Source Code

Here is source code of the C program to find the size of a Union. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to find the size of a union
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     union sample
9.     {
10.         int m;
11.         float n;
12.         char ch;
13.     };
14.     union sample u;
15.
16.     printf("The size of union = %d\n", sizeof(u));
17.     /* initialization */
18.     u.m = 25;
19.     printf("%d %f %c\n", u.m, u.n, u.ch);
20.     u.n = 0.2;
21.     printf("%d %f %c\n", u.m, u.n, u.ch);
22.     u.ch = 'p';
23.     printf("%d %f %c\n", u.m, u.n, u.ch);
24. }
```

Program Explanation

1. Define the union named sample.
2. Declare three variables m, n and ch of different data types.
3. Use the keyword sizeof() to find the size of a union and print the same.
4. Initialize each variable with some value and print its value as output.
5. Exit.

advertisement

Runtime Test Cases

The size of union = 4

25 0.000000

1045220557 0.200000

1045220464 0.199999

64. C Program to Accept an Array & Swap Elements using Pointers

[« Prev](#)

[Next »](#)

This is a C Program to accept an array & swap elements using pointers.

Problem Description

The program will implement an array and will swap the elements of the array. Swapping is done using pointers.

Problem Solution

1. Declare an array and define all its elements.
2. Create a function with two parameters i.e. two pointer variables.
3. Inside this function, first create a temporary variable. Then this temporary variable is assigned the value at first pointer.
4. Now, value at first pointer changes to the value at second pointer.
5. And value at second pointer changes to the value of temporary variable.
6. This way swapping is done and in main() function, we need to pass two pointer variables pointing the element which we need to swap.

advertisement

Program/Source Code

Here is source code of the C program to accept an array & swap elements using pointers. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C program to accept an array of 10 elements and swap 3rd element
3.  * with 4th element using pointers and display the results.
4. */
5.
6. #include <stdio.h>
7. void swap34(float *ptr1, float *ptr2);
8. void main()
9. {
10.
11.     float x[10];
12.     int i, n;
13.
14.     printf("How many Elements...\\n");
15.     scanf("%d", &n);
16.
17.     printf("Enter Elements one by one\\n");
18.     for (i = 0; i < n; i++)
19.     {
20.         scanf("%f", x + i);
21.     }
22.
23.     /* Function call:Interchanging 3rd element by 4th */
24.
25.     swap34(x + 2, x + 3);
26.     printf("\\nResultant Array...\\n");
```

```

27.
28.     for (i = 0; i < n; i++)
29.     {
30.         printf("X[%d] = %f\n", i, x[i]);
31.     }
32.
33. }
34.
35. /* Function to swap the 3rd element with the 4th element in the array */
36.
37. void swap34(float *ptr1, float *ptr2 )
38. {
39.
40.     float temp;
41.     temp = *ptr1;
42.     *ptr1 = *ptr2;
43.     *ptr2 = temp;
44.
45. }
```

Program Explanation

1. Declare an array and define all the elements according to its size taken from users.
2. Now create a function passing two pointer variables of float type as parameters to this function.
3. Inside a function a variable temp is declared of float type, which will store the value pointed by first pointer variable.
4. Now value pointer by first pointer variable is changed to the value pointed by second pointer variable, and value pointer by second variable is change to the value of temp variable.
5. Exit the function.
6. In the main() method, call this function with pointers pointing to those two element which you want to swap.

advertisement

Runtime Test Cases

How many Elements...

4

Enter Elements one by one

23

67

45

15

Resultant Array...

X[0] = 23.000000

X[1] = 67.000000

X[2] = 15.000000

X[3] = 45.000000

65. C Program to Cyclically Permute the Elements of an Array

[« Prev](#)

[Next »](#)

This is a C Program to cyclically permutes the elements of an array.

Problem Description

This program first accepts an array. Assume there are 4 elements in an array. It takes 2 element as a first element in an array and so on till the last element of the given array. Now here first element of an array becomes last element in an array during cyclical permutation.

Problem Solution

1. Create a one-dimentional array of some fixed size (lets say n), defining all its elements.
2. Reserve the first element of the array by assigning its value to the nth position of the array.
3. Now using for loop from 0 to size-1, with iterator i, each value at (i+1)th position is assigned to the ith position of array.
4. Because the nth position holds the value of 0th position, therefore the last element will have the value which was earlier the first element.

advertisement

Program/Source Code

Here is source code of the C program to cyclically permutes the elements of an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C program to cyclically permute the elements of an array A.
3.  * i.e. the content of A1 become that of A2. And A2 contains
4.  * that of A3 & so on as An contains A1
5. */
6.
7. #include <stdio.h>
8. void main ()
9. {
10.
11.    int i, n, number[30];
12.    printf("Enter the value of the n = ");
13.    scanf("%d", &n);
14.
15.    printf("Enter the numbers\n");
16.    for (i = 0; i < n; ++i)
17.    {
18.        scanf("%d", &number[i]);
19.    }
20.
21.    number[n] = number[0];
22.    for (i = 0; i < n; ++i)
23.    {
24.        number[i] = number[i + 1];
25.    }
26.
27.    printf("Cyclically permuted numbers are given below \n");
28.    for (i = 0; i < n; ++i)
29.        printf("%d\n", number[i]);
30.
```

Program Explanation

1. Create an array of integer of some certain maximum capacity (10, in this case).
2. From users, take a number N as input, which will indicate the number of elements in the array ($N \leq$ maximum capacity)
3. Iterating through for loops (from [0 to N]), take integers as input from user and print them. These inputs are the elements of the array.
4. Now assign the value of first element of the array at 0th position to the nth position of the array.
5. Starting a for loop, with i as iterator from 0 to size-1, assigning each element at $(i+1)$ th position to ith position, hence each element shifts left by one.
6. And value of last element at $(n-1)$ th position would be assigned a value at nth position. Remember we stored the very first value of array at nth position.

advertisement

Runtime Test Cases

Enter the value of the n = 4

Enter the numbers

3

40

100

68

Cyclically permuted numbers are given below

40

100

68

3

66. C Program to Illustrate the Concept of Unions

[« Prev](#)

[Next »](#)

This is a C program to illustrate the concept of unions.

Problem Description

This program illustrates the concept of unions.

Problem Solution

1. Define the union.
2. Take the input and store it in the variable using dot operator.
3. Print the output using dot operator and exit.

advertisement

Program/Source Code

Here is source code of the C program to illustrate the concept of unions. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to illustrate the concept of unions
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     union number
9.     {
10.         int n1;
11.         float n2;
12.     };
13.     union number x;
14.
15.     printf("Enter the value of n1: ");
16.     scanf("%d", &x.n1);
17.     printf("Value of n1 = %d", x.n1);
18.     printf("\nEnter the value of n2: ");
19.     scanf("%f", &x.n2);
20.     printf("Value of n2 = %f\n", x.n2);
21. }
```

Program Explanation

1. Define the union named number with two variables n1 and n2.
2. Define the union variable x.
3. Take the value of two variables using dot operator(i.e. x.n1, x.n2) as input.
4. Print the values of two variables using dot operator as output.

advertisement

Runtime Test Cases

```
Enter the value of n1: 10
Value of n1 = 10
```

Enter the value of n2: 50
Value of n2 = 50.000000

67. C Program to Delete the Specified Integer from an Array

[« Prev](#)

[Next »](#)

This is C Program to delete a specified integer from an array.

Problem Description

It implements one dimentional array, take a number as input from users to delete and delete that element from the array in case it is present or print appropriate message if not present.

Problem Solution

1. Create an array of some size, and fill up its elements.
2. Take a value as input from users, which needs to be deleted.
3. Using for loop, check whether the value is present in the array or not.
4. If the value is present, then save its location and if its not present, some appropriate message get printed.
5. Again, the loop runs from that saved position till the end of array, causing each element to shift left by one step.
6. This, way the value get deleted.

advertisement

Program/Source Code

Here is source code of the C program to delete the specified integer from an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C program to accept an array of integers and delete the
3.  * specified integer from the list
4.  */
5.
6.
7.
8.
9. #include <stdio.h>
10. void main()
11. {
12.     int vectorx[10];
13.     int i, n, pos, element, found = 0;
14.
15.     printf("Enter how many elements\n");
16.     scanf("%d", &n);
17.     printf("Enter the elements\n");
18.
19.     for (i = 0; i < n; i++)
20.     {
21.         scanf("%d", &vectorx[i]);
22.     }
23.
24.     printf("Input array elements are\n");
25.     for (i = 0; i < n; i++)
26.     {
27.         printf("%d\n", vectorx[i]);
28.     }
29.
```

```

30. printf("Enter the element to be deleted\n");
31. scanf("%d", &element);
32.
33. for (i = 0; i < n; i++)
34. {
35.     if (vectorx[i] == element)
36.     {
37.         found = 1;
38.         pos = i;
39.         break;
40.     }
41. }
42.
43. if (found == 1)
44. {
45.     for (i = pos; i < n - 1; i++)
46.     {
47.         vectorx[i] = vectorx[i + 1];
48.     }
49.
50.     printf("The resultant vector is \n");
51.     for (i = 0; i < n - 1; i++)
52.     {
53.         printf("%d\n", vectorx[i]);
54.     }
55.
56. }
57. else
58.     printf("Element %d is not found in the vector\n", element);
59.
60. }

```

Program Explanation

1. Declare an array, vectorx of some fixed capacity, 10.
2. Take size of the array as input from users.
3. Using for loop, define the elements of the array.
4. Now, take a number form users as input, which needs to be deleted.
5. Run a for loop, comparing each element of the array to that number if both have same magnitude.
6. If the number is present in the array, then save its location. If number is not present in the array, then print appropriate message.
7. Run a for loop from that saved location to the size of array, shifting each element of the array leftwards by one.
8. This way, the number gets deleted.
9. Exit

advertisement

Runtime Test Cases

Enter how many elements

4

Enter the elements

345

234

678

987

Input array elements are

345

234

678

987

Enter the element to be deleted

234

The resultant vector is

345

678

987

68. C Program to Split an Array from Specified Position & Add First Part to the End

[« Prev](#)

[Next »](#)

This is a C Program to Split an Array from Specified Position & Add First Part to the End.

Problem Description

This program will split an array from specified position & add first part to the end. This program first accepts an array. Then splits an array according to the user specification. Now it becomes 2 parts & then add first part of an array at the end of a second part.

Problem Solution

1. Create an array of some certain size and define its elements in sorted fashion.
2. Take the position from users as input from where you want to split the array.
3. Run a for loop the position entered by user times, in which the element is one by one shifted towards left, and the leftmost element (i.e the first element of the array) is shifted to the nth position of the array.
4. One by one leftwards shifting is again done by a loop from 0 to n(size), where the value at (i+1)th position is assigned to the ith position of array. Thus, the element at nth position (where we stored value of 0th position) is assigned to (n-1)th position of array.

advertisement

Program/Source Code

Here is source code of the C program to split an array from specified position & add first part to the end. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C program to read an array, accept a key & split the array.
3.  * Add the first half to the end of second half.
4. */
5.
6.
7. #include <stdio.h>
8. void main ()
9. {
10.
11.    int number[30];
12.    int i, n, a, j;
13.
14.    printf("Enter the value of n\n");
15.    scanf("%d", &n);
16.
17.    printf("enter the numbers\n");
18.    for (i = 0; i < n; ++i)
19.        scanf("%d", &number[i]);
20.
21.    printf("Enter the position of the element to split the array \n");
22.    scanf("%d", &a);
23.
24.    for (i = 0; i < a; ++i)
25.    {
```

```

26.
27.     number[n] = number[0];
28.     for (j = 0; j < n; ++j)
29.     {
30.         number[j] = number[j + 1];
31.     }
32.
33. }
34.
35. printf("The resultant array is\n");
36.
37. for (i = 0; i < n; ++i)
38. {
39.     printf("%d\n", number[i]);
40. }
41.
42. }
```

Program Explanation

1. Declare an array of capacity 20, taking size from users, define all the element of the array but in sorted fashion.
2. Take from users as input the position from where you want to split the array.
3. Now, start a nested for loop, where the outer loop runs the position entered by user times. At each turn the array elements are shifted towards left, shifting the leftmost array element to the nth position of the array (where n is the size of the array)
4. The inner loop shifts all the array elements leftwards one by one by assigning the element at (i+1)th position to the ith position, keeping the element at 0th position to the nth position, so that after this loop ends, the first element of the array becomes the last element.
5. Executing this nested loop will provide us with an array having split up by the position entered by user.
6. Print the array and exit.

advertisement

Runtime Test Cases

```

Enter the value of n
4
enter the numbers
3
678
345
876
Enter the position of the element to split the array
3
The resultant array is
876
3
678
345
```

69. C Program to Generate Pascal Triangle 1 D Array

[« Prev](#)

[Next »](#)

This C Program generates pascal triangle 1 dimensional array.

Here is source code of the C Program to generate pascal triangle 1 dimensional array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Generate Pascal Triangle 1 D Array
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int array[30], temp[30], i, j, k, l, num;      //using 2 arrays
9.
10.    printf("Enter the number of lines to be printed: ");
11.    scanf("%d", &num);
12.    temp[0] = 1;
13.    array[0] = 1;
14.    for (j = 0; j < num; j++)
15.        printf(" ");
16.    printf(" 1\n");
17.    for (i = 1; i < num; i++)
18.    {
19.        for (j = 0; j < i; j++)
20.            printf(" ");
21.        for (k = 1; k < num; k++)
22.        {
23.            array[k] = temp[k - 1] + temp[k];
24.        }
25.        array[i] = 1;
26.        for (l = 0; l <= i; l++)
27.        {
28.            printf("%3d", array[l]);
29.            temp[l] = array[l];
30.        }
31.        printf("\n");
32.    }
33. }
```

advertisement

```
$ cc pgm69.c
$ a.out
```

Enter the number of lines to be printed: 4

```
 1
 1 1
 1 2 1
 1 3 3 1
```

70. C Program to Print the Number of Odd & Even Numbers in an Array

[« Prev](#)

[Next »](#)

This is a C Program to print the number of odd & even numbers in an Array.

Problem Description

This C Program prints the number of odd & even numbers in an array.

Problem Solution

1. Create an array, take its size from users and define its elements using a loop.
2. Take an iterator in a for loop, using which, all the elements of the array are accessed.
3. Iterator is used to reach out every position of the array, scanning the particular array element and checking whether it is divisible by 2 or not, thus sorting even and odd numbers and printing them.

advertisement

Program/Source Code

Here is source code of the C Program to print the number of odd & even numbers in an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to Print the Number of Odd & Even Numbers in an Array
3.  */
4.
5. #include <stdio.h>
6. void main()
7. {
8.
9.     int array[100], i, num;
10.    printf("Enter the size of an array \n");
11.
12.    scanf("%d", &num);
13.    printf("Enter the elements of the array \n");
14.
15.    for (i = 0; i < num; i++)
16.    {
17.        scanf("%d", &array[i]);
18.    }
19.
20.    printf("Even numbers in the array are - ");
21.    for (i = 0; i < num; i++)
22.    {
23.        if (array[i] % 2 == 0)
24.        {
25.            printf("%d \t", array[i]);
26.        }
27.    }
28.
29.    printf("\n Odd numbers in the array are - ");
30.    for (i = 0; i < num; i++)
31.    {
32.        if (array[i] % 2 != 0)
33.        {
34.            printf("%d \t", array[i]);
35.        }
36.    }
```

```
36.    }
37.
38. }
```

Program Explanation

1. Declare an array of some certain capacity, 100. Take size from users and define its elements.
2. Using for loop, taking iterator i, to reach every position of array and accessing every element of it, check whether the element is divisible by 2 or not.
3. If the element modulo 2 is 0, that means the number is divisible by 2, hence even, print it out.
4. Similarly if the element modulo 2 is 1, that means the number is not divisible by 2, hence odd, print it out.

advertisement

Runtime Test Cases

Enter the size of an array

6

Enter the elements of the array

12

19

45

69

98

23

Even numbers in the array are - 12 98

Odd numbers in the array are - 19 45 69 23

71. C Program to Print all the Repeated Numbers with Frequency in an Array

[« Prev](#)

[Next »](#)

This is a C Program to print all the repeated numbers with frequency in an array.

Problem Description

This C Program print all the repeated numbers with frequency in an array.

Problem Solution

Prints the repeated numbers with frequency in a given array as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to print all the repeated numbers with frequency in an array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Print all the Repeated Numbers with Frequency in an Array
 */
#include <stdio.h>
#include <malloc.h>

void duplicate(int array[], int num)
{
    int *count = (int *)calloc(sizeof(int), (num - 2));
    int i;

    printf("duplicate elements present in the given array are ");
    for (i = 0; i < num; i++)
    {
        if (count[array[i]] == 1)
            printf(" %d ", array[i]);
        else
            count[array[i]]++;
    }
}

int main()
{
    int array[] = {5, 10, 10, 2, 1, 4, 2};
    int array_freq = sizeof(array) / sizeof(array[0]);
    duplicate(array, array_freq);
    getchar();
    return 0;
}
```

Program Explanation

In this C Program, we have defined the array[] variable elements. Compute the division of size of previous array value by the next array[] variable element value. the duplicate() function is used to find the repeated numbers with frequency in an array.

In duplicate() function using for loop find the duplicate numbers present in the given array. If else condition statement is used to check that count[] array variable with base index value of array[] variable is equal to 1.

If the condition is true, then execute the statement and print the duplicate elements in the array. Otherwise, if the condition is false, then execute the else statement and increment the count variable value.

Runtime Test Cases

```
$ cc pgm71.c
$ a.out
duplicate elements present in the given array are 10 2
```

72. C Program to Print the kth Element in the Array

[« Prev](#)

[Next »](#)

This is a C program to print the kth element in the array.

Problem Description

This C Program implements an array prints the kth element in the array where the position k is entered by the user as an input.

Problem Solution

1. Create an array and taking its size from the users, define all its elements.
2. Take an input from users, the position in the array where we want to access element.
3. The element would be in the index (entered_position -1) of the array, print it.

advertisement

Program/Source Code

Here is source code of the C Program to print the kth element in the array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to Print the kth Element in the Array
3.  */
4.
5. #include <stdio.h>
6. int main()
7. {
8.     int arr[100], len, i, j, temp, n;
9.     printf("Enter the size of array");
10.
11.    scanf("%d", &len);
12.    printf("\n Enter the array elements");
13.
14.    for (i = 0; i < len; i++)
15.    {
16.        scanf("%d", &arr[i]);
17.    }
18.
19.    printf("\n Enter Which kth Number You want");
20.    scanf("%d", &n);
21.    printf("\n The %d th kth number is: %d", n, arr[n - 1]);
22.    return 0;
23.
24. }
```

Program Explanation

1. Declare an array of some fixed capacity, 100.
2. Take size from users as an input.
3. Using for loop, define the elements of the array according to the size.
4. Enter the position(i.e k) where we want to access the element.
5. The index of the array where we will find the required element according to the position entered is – (k-1).

advertisement

Runtime Test Cases

Enter the size of array4

Enter the array elements

12

13

17

20

Enter Which kth Number You want 4

The 4th kth number is: 20

73. C Program to Find the Largest Number in an Array

[« Prev](#)

[Next »](#)

This is a C Program to find the largest number in an array.

Problem Description

We have to write a program in C such that the program will read a one-dimensional array and find out the largest element present in the array.

Expected Input and Output

If we are entering 5 elements ($N = 5$), with array element values as 12, 56, 34, 78 and 100

Then, **largest element present in the given array is: 100**
advertisement

Problem Solution

Fundamentally, an array is a data structure containing a collection of values or variables. The simplest type of array is a linear array or one-dimensional array. An array can be defined in C with the following syntax:

```
int Arr[5] = {12, 56, 34, 78, 100};  
/* here 12,56,34,78,100 are the elements at indices 0,1,2,3,4 respectively */
```

In this example, array **Arr** is a collection of 5 integers. Each integer can be identified and accessed by its index. The indices of the array start with 0, so the first element of the array will have index 0, the next will have index 1 and so on.

advertisement

In this program, we have to find the largest element present in the array. We will do this by first saving the value of the first element in the variable '**largest**'. Then we will compare with remaining elements of the array and store the value if another larger number is found in this array. This will go on $N-1$ times and the program ends. The sequence of steps for the solution will be as follows:

1. Create an array of user-defined size.
2. Run the for loop till the user-defined size to insert the element at each location.
3. Considering the first element of the array to be the largest, compare all the remaining elements of the array, and change the largest value if assumed largest element is smaller than the element being compared.
4. At last, the largest element will hold the actual largest value in the array. Thus, print it.

Program/Source Code

Here is the source code of the C Program to find the largest number in an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

advertisement

```
1. /*  
2.  * C program to read N integers into an array A and  
3.  * a) Find the sum of all numbers  
4.  * b) Find the average of all numbers  
5.  * Display the results with suitable headings  
6. */  
7.  
8. #include <stdio.h>  
9.  
10.int main()  
11.{  
12.  
13.    int size, i, largest;  
14.  
15.    printf("\n Enter the size of the array: ");  
16.    scanf("%d", &size);  
17.    int array[size];  
18.
```

```

19.     printf("\n Enter %d elements of the array: \n", size);
20.
21.     for (i = 0; i < size; i++)
22.     {
23.         scanf("%d", &array[i]);
24.     }
25.
26.     largest = array[0];
27.
28.     for (i = 1; i < size; i++)
29.     {
30.         if (largest < array[i])
31.             largest = array[i];
32.     }
33.
34.     printf("\n largest element present in the given array is : %d", largest);
35.
36.     return 0;
37.
38.

```

Program Explanation

1. Take the size of the array as input from the user.
2. Then, initialize an array of size given by the user.
3. Using **for** loop, take array element as input from users and insert them into the array.
4. After inserting all the elements of the array, consider the very first element of array to be the **largest**.
5. Run a for loop, from 1 to arraySize-1, extracting array element one by one and comparing it to the **largest** element.
6. If the **largest** element is smaller than the element being compared, then the largest element is updated with the value of the current element of the array.
7. In the end, the **largest** element will hold the actual largest value present in the array.

Runtime Test Cases

Here is the runtime output of the C program where the user is reading array of 5 elements with values as 12, 56, 34, 78 and 100. Then it finds out the largest element and displays its value.

advertisement

Enter the size of the array: 5

Enter 5 elements of the array:

12
56
34
78
100

largest element present in the given array is: 100

74. C Program to Find the Number of Elements in an Array

[« Prev](#)

[Next »](#)

This is a C Program to find the number of elements in an array.

Problem Description

This C Program finds the number of elements present in the given array.

Problem Solution

1. Create an array along with the definition of all its elements.
2. Create a variable which will hold the size of the array.
3. Using the inbuilt function sizeof(), passing the name of the array whose size we need to calculate, returns the number of elements contained by array.
4. The returned value is stored in the variable created above.

advertisement

Program/Source Code

Here is source code of the C Program to find number of elements present in the given array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.  
2. /*  
3.  * C Program to Find the Number of Elements in an Array  
4. */  
5.  
6. #include <stdio.h>  
7. #include <stdlib.h>  
8. #include <unistd.h>  
9.  
10. int main()  
11. {  
12.     int array[] = {15, 50, 34, 20, 10, 79, 100};  
13.     int n;  
14.     n = sizeof(array);  
15.     printf("Size of the given array is %d\n", n/sizeof(int));  
16.     return 0;  
17. }
```

Program Explanation

1. Declare an array and define its elements at the time of declaration.
2. Create a variable n, which will store the size of the array, i.e total number of elements present in array.
3. Now, call an inbuilt function sizeof() (from unistd.h library) passing the name of the array as its parameter.
4. This function will return the number of elements present in the array passed to it.
5. Store it in variable n.

Runtime Test Cases

Size of the given array is 7

75. C Program to Convert Binary to Octal

[« Prev](#)

[Next »](#)

This is a C program to Convert Binary to octal.

Problem Description

This program takes a binary number as input and converts to octal.

Problem Solution

1. Take a binary number as input.
2. Divide the binary number into groups of 3 bits. For each group of 3 bits, multiply each bit with the power of 2 and add them consecutively.
3. Combine the result of all groups to get the output.

advertisement

Program/Source Code

Here is source code of the C program to Convert Binary to Octal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Convert Binary to Octal
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     long int binarynum, octalnum = 0, j = 1, remainder;
9.
10.    printf("Enter the value for binary number: ");
11.    scanf("%ld", &binarynum);
12.    while (binarynum != 0)
13.    {
14.        remainder = binarynum % 10;
15.        octalnum = octalnum + remainder * j;
16.        j = j * 2;
17.        binarynum = binarynum / 10;
18.    }
19.    printf("Equivalent octal value: %lo", octalnum);
20.    return 0;
21.}
```

Program Explanation

1. Take a binary number as input and store it in the variable binarynum.
2. Obtain the remainder and quotient of the input number by dividing it by 10.
3. Multiply the obtained remainder with variable j and increment the variable octalnum with this value.
4. Increment the variable j by 2 and override the variable binarynum with the quotient obtained.
5. Repeat the steps 2-4 until the variable binarynum becomes zero.
6. Print the variable octalnum as output.

advertisement

Runtime Test Cases

Output:

```
Enter the value for binary number: 10101
Equivalent octal value: 25
```

76. C Program to Convert Binary to Hexadecimal

[« Prev](#)

[Next »](#)

This is a C program to Convert Binary to Hexadecimal.

Problem Description

This program takes a binary number as input and converts to hexadecimal.

Problem Solution

1. Take a binary number as input.
2. Divide the binary number into groups of 4 bits. For each group of 4 bits, multiply each bit with the power of 2 and add them consecutively.
3. Combine the result of all groups to get the output.

advertisement

Program/Source Code

Here is source code of the C program to Convert Binary to Hexadecimal . The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Convert Binary to Hexadecimal
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     long int binaryval, hexadecimalval = 0, i = 1, remainder;
9.
10.    printf("Enter the binary number: ");
11.    scanf("%ld", &binaryval);
12.    while (binaryval != 0)
13.    {
14.        remainder = binaryval % 10;
15.        hexadecimalval = hexadecimalval + remainder * i;
16.        i = i * 2;
17.        binaryval = binaryval / 10;
18.    }
19.    printf("Equivalent hexadecimal value: %lx", hexadecimalval);
20.    return 0;
21.}
```

Program Explanation

1. Take a binary number as input and store it in the variable binaryval.
2. Obtain the remainder and quotient of the input number by dividing it by 10.
3. Multiply the obtained remainder with variable i and increment the variable hexadecimalval with this value.
4. Increment the variable i by 2 and override the variable binaryval with the quotient obtained.
5. Repeat the steps 2-4 until the variable binaryval becomes zero.
6. Print the variable hexadecimalval as output.

advertisement

Runtime Test Cases

Output:

```
Enter the binary number: 10000
Equivalent hexadecimal value: 10
```

77. C program to Convert Decimal to Octal

[« Prev](#)

[Next »](#)

This is a C program to Convert Decimal to Octal.

Problem Description

This program takes a decimal number as input and converts to octal number.

Problem Solution

1. Take a decimal number as input.
2. Divide the input number by 8 and obtain its remainder and quotient. Store the remainder in the array.
3. Repeat the step 2 with the quotient obtained. Do this until the quotient becomes zero.
4. Print the array in the reverse order to get the output.

advertisement

Program/Source Code

Here is source code of the C program to Convert Decimal to Octal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to Convert Decimal to Octal
3.  */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     long decimalnum, remainder, quotient;
9.     int octalNumber[100], i = 1, j;
10.
11.    printf("Enter the decimal number: ");
12.    scanf("%ld", &decimalnum);
13.    quotient = decimalnum;
14.    while (quotient != 0)
15.    {
16.        octalNumber[i++] = quotient % 8;
17.        quotient = quotient / 8;
18.    }
19.    printf("Equivalent octal value of decimal no %d: ", decimalnum);
20.    for (j = i - 1; j > 0; j--)
21.        printf("%d", octalNumber[j]);
22.    return 0;
23.}
```

Program Explanation

1. Take a decimal number as input and store it in the variable decimalnum.
2. Copy the variable decimalnum to the variable quotient.
3. Divide the variable quotient and obtain its remainder and quotient. Store the remainder in the array octalNumber and override the variable quotient with the quotient obtained.
4. Repeat the step 3 until the quotient becomes zero.
5. When it becomes zero, print the array octalNumber in the reverse order to get the output.

advertisement

Runtime Test Cases

Output:

Enter the decimal number: 68

Equivalent octal value of decimal no 68: 104

78. C program to Convert Decimal to Hexadecimal

[« Prev](#)

[Next »](#)

This is a C program to Convert Decimal to Hexadecimal.

Problem Description

This program takes a decimal number as input and converts to hexadecimal.

Problem Solution

1. Take a decimal number as input.
2. Divide the input number by 16. Store the remainder in the array.
3. Do step 2 with the quotient obtained until quotient becomes zero.
4. Print the array in the reversed fashion to get hexadecimal number.

advertisement

Program/Source Code

Here is source code of the C program to Convert Decimal to Hexadecimal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C program to Convert Decimal to Hexadecimal
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     long decimalnum, quotient, remainder;
9.     int i, j = 0;
10.    char hexadecimalnum[100];
11.
12.    printf("Enter decimal number: ");
13.    scanf("%ld", &decimalnum);
14.
15.    quotient = decimalnum;
16.
17.    while (quotient != 0)
18.    {
19.        remainder = quotient % 16;
20.        if (remainder < 10)
21.            hexadecimalnum[j++] = 48 + remainder;
22.        else
23.            hexadecimalnum[j++] = 55 + remainder;
24.        quotient = quotient / 16;
25.    }
26.
27. // display integer into character
28. for (i = j; i >= 0; i--)
29.     printf("%c", hexadecimalnum[i]);
30. return 0;
31. }
```

Program Explanation

1. Take a decimal number as input and store it in the variable decimalnum.
2. Initialize the variable j=0 and copy decimalnum to variable quotient.
3. Obtain the quotient and remainder of the variable quotient. Store the obtained remainder in the variable remainder and override the variable quotient with obtained quotient.
4. Check if the remainder is less than 10. If it is, then add 48 to the remainder and store the result in the array hexadecimalnum. Otherwise, add 55 to the remainder and store the result in the array hexadecimalnum.
5. Do steps 3-4 until variable quotient becomes zero.
6. When it becomes zero, print the array hexadecimalnum in the reversed fashion as output.

advertisement

Runtime Test Cases

Output:

```
Enter decimal number: 12
Equivalent hexadecimal value of 12 : C
```

79. C Program to Find the Sum of Contiguous Subarray within a 1 – D Array of Numbers which has the Largest Sum

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of contiguous subarray within a 1 – D array of numbers which has the largest sum.

Problem Description

We have to write a program in C such that the program will find the sum of contiguous subarray within a 1 – D array of numbers (one-dimensional array of numbers) which has the largest sum.

Suppose, we have an array of 8 elements with values: -1,-5,5,3,-2,5,4 and 1, then here is a sample of various possible contiguous subarrays:

advertisement

-1
-1,-5
-1,-5,5
-1,-5,5,3
-1,-5,5,3,-2
-1,-5,5,3,-2,5
-1,-5,5,3,-2,5,4
-1,-5,5,3,-2,5,4,1

-5

-5,5

-5,5,3

....

....

....

For each subarray, we have to do the sum of the elements of the subarray and then find the subarray which has the largest sum.

Expected Input and Output

If we are entering 8 elements ($N = 8$), with array element values as -1,-5,5,3,-2,5,4 and 1 then,

The largest contiguous subarray is: 5 3 -2 5 4 1

The sum of the largest contiguous subarray is: 16

Problem Solution

In this program, we will print the contiguous subarray within one dimensional array of numbers which has the largest sum.

advertisement

We will do this by iterating over every possible contiguous combination of the array using 2 **for** loops.

Then we will compare them to a variable **largest** which is initialized with a value of first element of the array, say **array[0]**. For every contiguous subarray, we will add the elements of that subarray and then compare it with the variable **largest** to find the largest sum and also store the address of the **starting and ending index**. In the end, we will print the Largest sum and the corresponding subarray.

advertisement

The sequence of steps for the solution will be as follows:

1. Create an array of user-defined size.
2. Run a **for** loop to read the elements of the array.
3. Considering the first element of the array to be the largest, compare all the contiguous subarray sums, and change the largest value if the largest element is smaller than the current subarray sum.
4. At last, the largest element will hold the actual largest contiguous subarray sum and then print it.

Program/Source Code

Here is the source code of the C Program to find the sum of contiguous subarray within a 1 – D array of numbers which has the largest sum. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Sum of Contiguous Subarray within a
3. * 1 - D Array of Numbers which has the Largest Sum
4. */
5.
6. #include<stdio.h>
7.
8. int main()
9. {
10.     int size,m=0,l=0;
11.
12.     printf("Type the length of the array\n");
13.     scanf("%d",&size);
14.     int array[size];
15.     printf("type the elements of the array\n");
16.
17.     for(int i=0;i<size;i++)
18.     {
19.         scanf("%d",&array[i]);
20.
21.     }
22.
23.     int largest=array[0];
24.     for(int i=0;i<size;i++)
25.     {
26.         int sum=0;
27.         for(int j=i;j<size;j++)
28.         {
29.             sum=sum+array[j];
30.             if(sum>largest)
31.             {
32.                 m=i;l=j;
33.                 largest=sum;
34.             }
35.         }
36.     }
37.
38.     printf("\n The largest contigous subarray is");
39.     for(int z=m;z<=l;z++)
40.     {
41.         printf(" %d ",array[z]);
42.     }
43.     printf("\n The sum of the largest contigous subarray is");
44.     printf(" %d",largest);
```

```
45. return 0;  
46. }
```

Program Explanation

1. Take the size of the array as input from users.
2. Then, Initialize an array of size given by the user.
3. Using for loop, take array element as input from users and insert them into the array.
4. After inserting all the elements of the array, consider the very first element of array to be the largest.
5. Run a for loop, from 1 to arraySize-1, extracting array element one by one.
6. Run another loop inside this loop and sum every possible contiguous subarray.
7. If the largest element is smaller than the sum of the current contiguous subarray, then the largest element is updated to the current sum.
7. In the end, the largest element will hold the actual largest sum.

advertisement

Runtime Test Cases

Here is the runtime output of the C program where the user is reading an array of 8 elements with values as -1,-5,5,3,-2,5,4 and 1 and then it displays the largest contiguous subarray with its sum.

Type the length of the array

8

type the elements of the array

-1

-5

5

3

-2

5

4

1

The largest contiguous subarray is 5 3 -2 5 4 1

The sum of the largest contiguous subarray is 16

80. C Program to Find Area of Rhombus

[« Prev](#)

[Next »](#)

This is a C Program to find area of rhombus.

Problem Description

This C Program calculates the area of Rhombus.

Problem Solution

The formula used in this program are Area= (1/2) * height * width.

advertisement

Program/Source Code

Here is source code of the C Program to Find the area of rhombus.The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Area of rhombus
 */
#include <stdio.h>

int main()
{
    float diagonal1, diagonal2;
    float area;

    printf("Enter diagonals of the given rhombus: \n ");
    scanf("%f%f", &diagonal1, &diagonal2);
    area = 0.5 * diagonal1 * diagonal2;
    printf("Area of rhombus is: %.3f\n", area);
    return 0;
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the diagonal value of the rhombus using ‘diagonal1’ and ‘diagonal2’ variables. To find the area of a rhombus, the following formula is used.

$$\text{Area} = 0.5 * \text{diagonal1} * \text{diagonal2}$$

advertisement

Runtime Test Cases

Output:

\$ cc pgm26.c

\$ a.out

Enter diagonals of the given rhombus:

30 40

Area of rhombus is: 600.000

81. C Program to Convert Hexadecimal to Binary

[« Prev](#)

[Next »](#)

This is a C program to Convert Hexadecimal to Binary.

Problem Description

This program takes a hexadecimal number and converts to binary number.

Problem Solution

1. Take a hexadecimal number as input.
2. For each bit of a hexadecimal number print its equivalent binary number in a four bit fashion. Example: For 22 print it as 0010 0010.
3. Use switch statement to access each bit of a hexadecimal number.

advertisement

Program/Source Code

Here is source code of the C program to Convert Hexadecimal to Binary. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Convert Hexadecimal to Binary
3. */
4. #include <stdio.h>
5. #define MAX 1000
6.
7. int main()
8. {
9.     char binarynum[MAX], hexa[MAX];
10.    long int i = 0;
11.
12.    printf("Enter the value for hexadecimal ");
13.    scanf("%s", hexa);
14.    printf("\n Equivalent binary value: ");
15.    while (hexa[i])
16.    {
17.        switch (hexa[i])
18.        {
19.            case '0':
20.                printf("0000"); break;
21.            case '1':
22.                printf("0001"); break;
23.            case '2':
24.                printf("0010"); break;
25.            case '3':
26.                printf("0011"); break;
27.            case '4':
28.                printf("0100"); break;
29.            case '5':
30.                printf("0101"); break;
31.            case '6':
32.                printf("0110"); break;
33.            case '7':
34.                printf("0111"); break;
35.            case '8':
36.                printf("1000"); break;
37.            case '9':
38.                printf("1001"); break;
39.            case 'A':
40.                printf("1010"); break;
41.            case 'B':
42.                printf("1011"); break;
43.            case 'C':
44.                printf("1100"); break;
45.            case 'D':
46.                printf("1101"); break;
```

```

47.     case 'E':
48.         printf("1110"); break;
49.     case 'F':
50.         printf("1111"); break;
51.     case 'a':
52.         printf("1010"); break;
53.     case 'b':
54.         printf("1011"); break;
55.     case 'c':
56.         printf("1100"); break;
57.     case 'd':
58.         printf("1101"); break;
59.     case 'e':
60.         printf("1110"); break;
61.     case 'f':
62.         printf("1111"); break;
63.     default:
64.         printf("\n Invalid hexa digit %c ", hexa[i]);
65.         return 0;
66.     }
67.     i++;
68. }
69. return 0;
70.}

```

Program Explanation

1. Take a hexadecimal number as input and store it in the array hexa.
2. Using switch statement access each bit of the hexadecimal number and print its equivalent binary number in a four bit fashion as shown in the program.
3. Do step 2 for every bit of a input number. Use while loop to do this.

advertisement

Runtime Test Cases

Output:

Enter the value for hexadecimal ab
Equivalent binary value: 10101011

82. C Program to Display Pascal triangle

[« Prev](#)

[Next »](#)

This is a C Program to display pascal triangle.

Problem Description

This C Program displays pascal triangle.

Problem Solution

advertisement

Take input from the user and displays pascal triangle as shown in the program below.

Program/Source Code

Here is source code of the C Program to display Pascal triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Display Pascal triangle
 */
#include <stdio.h>

void main()
{
    int array[15][15], i, j, rows, num = 25, k;

    printf("\n enter the number of rows:");
    scanf("%d", &rows);
    for (i = 0; i < rows; i++)
    {
        for (k = num - 2 * i; k >= 0; k--)
            printf(" ");
        for (j = 0; j <= i; j++)
        {
            if (j == 0 || i == j)
            {
                array[i][j] = 1;
            }
            else
            {
                array[i][j] = array[i - 1][j - 1] + array[i - 1][j];
            }
            printf("%4d", array[i][j]);
        }
        printf("\n");
    }
}
```

Program Explanation

This C program is used to print the Pascal triangle. Pascal's triangle is a triangular array of the binomial coefficients. The program consists of six integer type of variable, named i, j, rows, array[][][], k and num.

advertisement

Out of these variable i,j and k have been defined to control the for() loop, the integer 'rows' stores the limit of Pascal's triangle entered by the user. As the program for Pascal's triangle is executed, it first asks for the value of limit of the triangle.

The program assigns 'rows' variable value with 'i' variable value, i.e., number of space with the limit of Pascal's triangle, for loop in which 'i' is the loop control variable. Again, in order to control the space, a nested for loop with 'k' as a control variable is used.

advertisement

Finally, for printing the elements in this program for Pascal's triangle in C, another nested for() loop of control variable 'j' has been used. The formula used to generate the numbers of

Pascal's triangle is: $\text{array}[i][j] = \text{array}[i - 1][j - 1] + \text{array}[i - 1][j]$

After printing one complete row of numbers of Pascal's triangle, the control comes out of the nested loops and goes to next line as commanded by '\n' code. The process repeats till the control number specified is reached.

Runtime Test Cases

advertisement

Output:
\$ cc pgm37.c
\$ a.out

enter the number of rows:2

```
    1
   1 1
```

83. C Program to Add two Complex Numbers

[« Prev](#)

[Next »](#)

This is a C Program to add two complex numbers.

Problem Description

This C Program adds two complex numbers.

Problem Solution

A complex number is a number that can be put in the form $a + bi$, where a and b are real numbers and i is called the imaginary unit, where $i^2 = -1$. In this expression, a is called the real part and b the imaginary part of the complex number.

advertisement

Program/Source Code

Here is source code of the C Program to add two complex numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Add two Complex Numbers
 */
#include <stdio.h>

struct complex
{
    int realpart, imaginary;
};

main()
{
    struct complex a, b, c;

    printf("Enter value of a and b complex number a + ib.\n");
    printf("value of complex number a is = ");
    scanf("%d", &a.realpart);
    printf("value of complex number b is = ");
    scanf("%d", &a.imaginary);
    printf("Enter value of c and d complex number c + id.\n");
    printf("value of complex number c is = ");
    scanf("%d", &b.realpart);
    printf("value of complex number d is = ");
    scanf("%d", &b.imaginary);
    c.realpart = a.realpart + b.realpart;
    c.imaginary = a.imaginary + b.imaginary;
    if (c.imaginary >= 0)
        printf("complex numbers sum is = %d + %di\n", c.realpart, c.imaginary);
    else
        printf("complex numbers sum = %d %di\n", c.realpart, c.imaginary);
    return 0;
}
```

Program Explanation

In this C program, we are reading the value for complex number using the ‘realpart’ and ‘imaginary’ variables respectively. A complex number is a number that can be put in the form $a + bi$, where ‘ a ’ and ‘ b ’ are real numbers and ‘ i ’ is called the imaginary unit, where $i^2 = -1$. In this expression, ‘ a ’ is called the real part and ‘ b ’ the imaginary part of the complex number.

advertisement

The variable ‘ a ’ and ‘ b ’ are the objects of struct complex and it is used to access the ‘realpart’ and ‘imaginary’ variables in struct complex. The ‘ $c.realpart$ ’ variable is used to add the value of $a.realpart$ and $b.realpart$ variables and ‘ $c.imaginary$ ’ variable is used to add the value of $a.imaginary$ and $b.imaginary$ variables. If-else condition statement is used to check the value of $c.imaginary$ variable is greater than or equal to 0, if the condition is true then it will execute the statement and print the value of addition of two complex numbers.

Runtime Test Cases

```
$ cc pgm55.c
$ a.out
Enter value of a and b complex number a + ib.
value of complex number a is = 10
value of complex number b is = 12
Enter value of c and d complex number c + id.
value of complex number c is = 15
value of complex number d is = 22
complex numbers sum is = 25 + 34i
```

84. C Program Delete a Specific Line from a Text File

[« Prev](#)

[Next »](#)

This is a C Program to delete a specific line from a text file.

Problem Description

This C Program deletes a specific line from a text file.

Problem Solution

Take input from the user and performs delete operations in text file as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to delete a specific line from a text file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program Delete a specific Line from a Text File
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     FILE *fileptr1, *fileptr2;
9.     char filename[40];
10.    char ch;
11.    int delete_line, temp = 1;
12.
13.    printf("Enter file name: ");
14.    scanf("%s", filename);
15.    //open file in read mode
16.    fileptr1 = fopen(filename, "r");
17.    ch = getc(fileptr1);
18.    while (ch != EOF)
19.    {
20.        printf("%c", ch);
21.        ch = getc(fileptr1);
22.    }
23.    //rewind
24.    rewind(fileptr1);
25.    printf("\n Enter line number of the line to be deleted:");
26.    scanf("%d", &delete_line);
27.    //open new file in write mode
28.    fileptr2 = fopen("replica.c", "w");
29.    ch = getc(fileptr1);
30.    while (ch != EOF)
31.    {
32.        ch = getc(fileptr1);
33.        if(ch == '\n')
34.            temp++;
35.        //except the line to be deleted
36.        if(temp != delete_line)
37.        {
38.            //copy all lines in file replica.c
39.            putc(ch, fileptr2);
40.        }
41.    }
42.    fclose(fileptr1);
43.    fclose(fileptr2);
44.    remove(filename);
45.    //rename the file replica.c to original name
46.    rename("replica.c", filename);
47.    printf("\n The contents of file after being modified are as follows:\n");
48.    fileptr1 = fopen(filename, "r");
49.    ch = getc(fileptr1);
50.    while (ch != EOF)
51.    {
52.        printf("%c", ch);
53.        ch = getc(fileptr1);
```

```
54. }
55. fclose(fileptr1);
56. return 0;
57.}
```

Program Explanation

This C Program, we are reading file name using ‘filename’ variable. Using ‘fileptr1’ variable Open the file in read mode. While loop is used to print the number of characters present in the file.

advertisement

Then rewind() function is used to set the file position to the beginning of the file of the given stream. Enter the line number of the line to be deleted using ‘delete_line’ variable.

Then ‘fileptr2’ variable is used to open the new file in write mode. While loop is used to print the number of characters present in the file. Is condition statement is used to copy except the line to be deleted. The file.Putc() function is used to copy all lines in file replica.c.

Then close the files and rename the file replica.c to original name. Using while loop print the contents of the file after being modified.

advertisement

Runtime Test Cases

```
$ cc pgm47.c
$ a.out
Enter file name: pgm1.c
/*
 * C PROGRAM TO CONVERSION FROM Decimal to hexadecimal
 */

#include<stdio.h>
int main()
{
    long int decimalnum, remainder, quotient;
    int i = 1, j, temp;
    char hexadecimalnum[100];

    printf("Enter any decimal number: ");
    scanf("%ld", &decimalnum);

    quotient = decimalnum;

    while (quotient != 0)
    {
        temp = quotient % 16;
        //To convert integer into character
        if (temp < 10)
            temp = temp + 48;
        else
            temp = temp + 55;

        hexadecimalnum[i++] = temp;
        quotient = quotient / 16;
    }

    printf("Equivalent hexadecimal value of decimal number %d: ", decimalnum);
    for (j = i - 1; j > 0; j--)
        printf("%c", hexadecimalnum[j]);
    return 0;
```

```
}
```

Enter line number of the line to be deleted: 10

The contents of file after being modified are as follows:

```
*  
* C PROGRAM TO CONVERSION FROM Decimal to hexadecimal  
*/  
  
#include<stdio.h>  
int main()  
{  
    long int decimalnum, remainder, quotient;  
    int i = 1, j, temp;  
  
    printf("Enter any decimal number: ");  
    scanf("%ld", &decimalnum);  
  
    quotient = decimalnum;  
  
    while (quotient != 0)  
    {  
        temp = quotient % 16;  
        //To convert integer into character  
        if (temp < 10)  
            temp = temp + 48;  
        else  
            temp = temp + 55;  
  
        hexadecimalnum[i++] = temp;  
        quotient = quotient / 16;  
    }  
  
    printf("Equivalent hexadecimal value of decimal number %d: ", decimalnum);  
    for (j = i - 1; j > 0; j--)  
        printf("%c", hexadecimalnum[j]);  
    return 0;  
}
```

85. C Program to Find the Volume and Surface Area of Cone

[« Prev](#)

[Next »](#)

This is a C Program to find the volume and surface area of cone.

Problem Description

This C Program calculates the volume and surface area of cone.

Problem Solution

This program is used to find the the volume and surface area of cone. The formula used in this program are Surface_area = $\text{Pi} * r * (r + \sqrt{r^2 + h^2})$, Volume = $\frac{1}{3} * \text{Pi} * r^2 * h$ where r is the radius and h is the height of the cone & Pi = 22/7.

advertisement

Program/Source Code

Here is source code of the C Program to Find the volume and surface area of cone. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the volume and surface area of cone
 */
#include <stdio.h>
#include <math.h>

int main()
{
    float radius, height;
    float surface_area, volume;

    printf("Enter value of radius and height of a cone :\n ");
    scanf("%f %f", &radius, &height);
    surface_area = (22 / 7) * radius * (radius + sqrt(radius * radius + height * height));
    volume = (1.0/3) * (22 / 7) * radius * radius * height;
    printf("Surface area of cone is: %.3f", surface_area);
    printf("\n Volume of cone is : %.3f", volume);
    return 0;
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the ‘radius’ and ‘height’ of a cone. To find the surface area and the volume, the following formulas are used.

advertisement

Surface area = $3.14 * \text{radius} * (\text{radius} + \sqrt{(\text{radius} * \text{radius}) + (\text{height} * \text{height})})$,

Volume = $\frac{1}{3} * 3.14 * \text{radius} * \text{radius} * \text{height}$.

Runtime Test Cases

```
Output:
$ cc pgm31.c -lm
$ a.out
Enter value of radius and height of a cone :
6 9
Surface area of cone is: 302.700
Volume of cone is : 324.000
```

86. C Program to Segregate 0s on Left Side & 1s on right side of the Array

[« Prev](#)

[Next »](#)

This C Program segregates 0s on left side & 1s on right side of the array.

Here is source code of the C Program to segregate 0s on left side & 1s on right side of the array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Segregate 0s on Left Side & 1s on right side of the Array (Traverse Array only once)
3. */
4. #include <stdio.h>
5.
6. /*Function to segregate all 0s on left and all 1s on right*/
7. void segregate0and1(int array[], int size)
8. {
9.     int left = 0, right = size-1;
10.
11.    while (left < right)
12.    {
13.        /* Increment left index while we see 0 at left */
14.        while (array[left] == 0 && left < right)
15.            left++;
16.        /* Decrement right index while we see 1 at right */
17.        while (array[right] == 1 && left < right)
18.            right--;
19.        /* If left is smaller than right then there is a 1 at left and a 0 at right. Exchange it */
20.        if(left < right)
21.        {
22.            array[left] = 0;
23.            array[right] = 1;
24.            left++;
25.            right--;
26.        }
27.    }
28.}
29.
30.int main()
31.{
32.    int arr[] = {0, 1, 0, 1, 1, 0};
33.    int array_size = 6, i = 0;
34.
35.    segregate0and1(arr, array_size);
36.    printf("segregated array is ");
37.    for (i = 0; i < 6; i++)
38.        printf("%d ", arr[i]);
39.    getchar();
40.    return 0;
41.

```

advertisement

```

$ cc pgm96.c
$ a.out
segregated array is 0 0 0 1 1 1

```

87. C Program to Input an Array, Store the Squares of these Elements in an Array & Print it

[« Prev](#)

[Next »](#)

C Program to Input an Array, Store the Squares of these Elements in an Array & Print it

Problem Description

This is a C program which inputs array & stores the squares of these elements in an array & print them.

Problem Solution

1. Create a two-dimentional array of define its elements statically.
2. Run a for loop, the number of rows in this two-dimentional array times.
3. Inside this for loop, a function is called passing every row of the 2D array as a parameter(i.e passing 1D array).
4. Inside this function, value of every column of that row is multiplied by its own number, and stored in the same place, thus storing the square of the previously stored number.
5. Now print all the elements of the 2D array.

advertisement

Program/Source Code

Here is source code of the C Program to input array & stores the of these elements in an array & print them. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to Input an Array, Store the Squares of these Elements in an Array & Print it
3. */
4.
5. #include <stdio.h>
6. #define MAX_ROWS 3
7. #define MAX_COLS 4
8.
9. void print_square(int [ ] );
10. void main (void)
11. {
12.
13.     int i;
14.     int num[MAX_ROWS][MAX_COLS] = { {10, 20, 30, 40}, {50, 60, 70, 80}, {90, 100, 110, 120} };
15.     for (i = 0; i < MAX_ROWS; i++)
16.         print_square(num[i]);
17.
18. }
19.
20. void print_square(int x[ ])
21. {
22.
23.     int j;
24.     for (j = 0; j < MAX_COLS; j++)
25.         printf ("%d\t", x[j] * x[j]);
26.     printf ("\n");
27.
28. }
```

Program Explanation

1. Declare a 2D array of some fixed row and column size (lets say 3 and 4 respectively).
2. Define all its elements at the time of declaration.
3. Run a for loop from 0 to row-1, sending each row of the 2D array in the function printSquare() as its parameter.
4. Inside this function, again a loop runs to access all the column values of the given row.
5. Thus the loop runs from 0 to column-1, accessing each column value, evaluating its square, storing the result at the same place.
6. Now, print the array.

advertisement

Runtime Test Cases

```
100 400 900 1600
2500 3600 4900 6400
8100 10000 12100 14400
```

\

88. C Program to Find the Sum of Series $1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of series $1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$.

Problem Description

This C Program calculates the Sum of Series $1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$.

Problem Solution

This program is used to find the sum of the given series.

advertisement

Program/Source Code

Here is source code of the C Program to Find the Sum of Series $1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find the Sum of Series 1 + 1/2 + 1/3 + 1/4 + ... + 1/N
 */
#include <stdio.h>

void main()
{
    double number, sum = 0, i;

    printf("\n enter the number ");
    scanf("%lf", &number);
    for (i = 1; i <= number; i++)
    {
        sum = sum + (1 / i);
        if (i == 1)
            printf("\n 1 +");
        else if (i == number)
            printf(" (1 / %lf)", i);
        else
            printf(" (1 / %lf) + ", i);
    }
    printf("\n The sum of the given series is %.2lf", sum);
}
```

Program Explanation

In this C Program, we are reading the limit to compute the summation from the series $1/1 + 2/2 + 3/3 + \dots + 1/N$ using ‘number’ integer variable.

advertisement

For loop is used to compute the summation of each integer value. Initialize the value of ‘i’ variable to 1. Check the condition that the value of ‘i’ variable is less than or equal to the value of ‘number’ variable. If the condition is true, then execute the iteration of the loop and add the sum of series.

Nested if else condition statement is used to check the value of ‘i’ variable is equal to 1. If the condition is true then execute the statement. Otherwise, if the condition is false, execute the else if statement. Check the condition that the value of ‘i’ variable is equal to the value of ‘number’ variable.

If the condition is true then execute the statement and compute the sum of series. Otherwise, if the condition is false, then execute else statement. Print the sum of series using printf statement.

advertisement

Runtime Test Cases

Output:
\$ cc pgm.c
\$ a.out

enter the number 4

$1 + (1/2.000000) + (1/3.000000) + (1/4.000000)$

The sum of the given series is 2.08

89. C Program to List Files in Directory

[« Prev](#)

[Next »](#)

This C Program lists files in directory.

Here is source code of the C Program to list files in directory. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to List Files in Directory
3. */
4. #include <dirent.h>
5. #include <stdio.h>
6.
```

```
7. int main(void)
8. {
9.     DIR *d;
10.    struct dirent *dir;
11.    d = opendir(".");
12.    if (d)
13.    {
14.        while ((dir = readdir(d)) != NULL)
15.        {
16.            printf("%s\n", dir->d_name);
17.        }
18.        closedir(d);
19.    }
20.    return(0);
21.}
```

advertisement

```
$ cc pgm59.c
```

```
$ a.out
```

```
.
```

```
..
```

```
b.txt
```

```
pgm2.c
```

```
pgm5.c
```

```
1
```

```
a.out
```

```
a.txt
```

```
b.txt
```

```
pgm.c
```

```
pgm1.c
```

```
pgm10.c
```

```
pgm11.c
```

```
pgm12.c
```

```
pgm13.c
```

```
pgm14.c
```

```
pgm15.c
```

```
pgm16.c
```

```
pgm17.c
```

```
pgm18.c
```

```
pgm19.c
```

```
pgm2.c
```

```
pgm20.c
```

```
pgm21.c
```

```
pgm22.c
```

```
pgm23.c
```

```
pgm24.c
```

```
pgm25.c
```

```
pgm26.c
```

```
pgm27.c
```

```
pgm28.c
```

```
pgm29.c
```

```
pgm3.c
```

```
pgm30.c
```

```
pgm31.c
```

```
pgm32.c
```

```
pgm33.c
```

```
pgm34.c
```

```
pgm35.c
```

```
pgm36.c
```

```
pgm37.c
```

```
pgm38.c
```

```
pgm39.c
```

```
pgm4.c
```

```
pgm40.c
```

```
pgm41.c
```

```
pgm42.c
```

```
pgm43.c  
pgm44.c  
pgm45.c  
pgm46.c  
pgm47.c  
pgm48.c  
pgm49.c  
pgm5.c  
pgm50.c  
pgm51.c  
pgm52.c  
pgm53.c  
pgm54.c  
pgm55.c  
pgm56.c  
pgm57.c  
pgm58.c  
pgm59.c  
pgm6.c  
pgm7.c  
pgm8.c  
pgm9.c
```

90. C Program to Display the ATM Transaction

[« Prev](#)

[Next »](#)

This is a C program to display the ATM transaction.

Problem Description

This C Program performs ATM transaction. The types of ATM transaction are

- 1) Balance checking
- 2) Cash withdrawal
- 3) Cash deposition.

Problem Solution

1. Firstly initialize the ATM pin and amount with some random number.
2. Take the ATM pin as input.
3. If the input pin is equal to the initialized pin, then do the further operations.
4. Use switch statement to do the operations like Balance checking, Cash withdrawal, Cash deposition etc.
5. Use while loop to terminate or restart the process.

advertisement

Program/Source Code

Here is source code of the C Program to display the ATM transaction. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Display the ATM Transaction
3. */
4. #include <stdio.h>
5.
6. unsigned long amount=1000, deposit, withdraw;
7. int choice, pin, k;
8. char transaction ='y';
9.
10.void main()
11.{ 
12.    while (pin != 1520)
13.    {
14.        printf("ENTER YOUR SECRET PIN NUMBER:");
15.        scanf("%d", &pin);
16.        if (pin != 1520)
17.            printf("PLEASE ENTER VALID PASSWORD\n");
18.    }
19.    do
20.    {
21.        printf("*****Welcome to ATM Service*****\n");
22.        printf("1. Check Balance\n");
23.        printf("2. Withdraw Cash\n");
24.        printf("3. Deposit Cash\n");
25.        printf("4. Quit\n");
26.        printf("*****?*****?\n\n");
27.        printf("Enter your choice: ");
28.        scanf("%d", &choice);
29.        switch (choice)
30.        {
31.            case 1:
32.                printf("\n YOUR BALANCE IN Rs : %lu ", amount);
33.                break;
34.            case 2:
35.                printf("\n ENTER THE AMOUNT TO WITHDRAW: ");
36.                scanf("%lu", &withdraw);
37.                if (withdraw % 100 != 0)
38.                {
39.                    printf("\n PLEASE ENTER THE AMOUNT IN MULTIPLES OF 100");
40.                }
41.                else if (withdraw >(amount - 500))
42.                {
43.                    printf("\n INSUFFICIENT BALANCE");
44.                }
45.                else
46.                {
47.                    amount = amount - withdraw;
48.                    printf("\n\n PLEASE COLLECT CASH");
49.                    printf("\n YOUR CURRENT BALANCE IS%lu", amount);
```

```

50.         }
51.         break;
52.     case 3:
53.         printf("\n ENTER THE AMOUNT TO DEPOSIT");
54.         scanf("%lu", &deposit);
55.         amount = amount + deposit;
56.         printf("YOUR BALANCE IS %lu", amount);
57.         break;
58.     case 4:
59.         printf("\n THANK U USING ATM");
60.         break;
61.     default:
62.         printf("\n INVALID CHOICE");
63.     }
64.     printf("\n\n\n DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n): \n");
65.     fflush(stdin);
66.     scanf("%c", &transaction);
67.     if (transaction == 'n' || transaction == 'N')
68.         k = 1;
69. } while (!k);
70. printf("\n\n THANKS FOR USING OUT ATM SERVICE");
71.

```

Program Explanation

1. Initialize the variables pin, amount and transaction with 1520, 1000 and ‘y’ respectively.
2. Ask for the pin from user. If the input pin is equal to 1520, then allow for the further operations.
3. Use switch statement to do the operations like Check Balance, Withdraw Cash, Deposit Cash and Quit.
4. For Check Balance simply print the variable amount as output and exit.
5. For Withdraw Cash, first ask the amount to withdraw and store it in the variable withdraw.
6. If withdraw % 100 != 0, then ask user to enter the amount in multiples of 100.
7. If withdraw amount is greater than (amount-500), then print the output as “INSUFFICIENT BALANCE”.
8. Otherwise subtract the variable withdraw from variable amount, print the amount and exit.
9. For deposit operation, ask the user for amount and store it in the variable deposit.
10. Add the variable deposit to variable amount, print the amount and exit.
11. If quit, then finally ask the user if they wish to continue or not. Ask them to type y/n and store it in the variable transaction.
12. If variable transaction is y/Y, then continue the operation. Otherwise terminate the while loop by assigning 1 to variable k.

advertisement

Runtime Test Cases

```

ENTER YOUR SECRET PIN NUMBER:1520
*****Welcome to ATM Service*****
1. Check Balance
2. Withdraw Cash
3. Deposit Cash
4. Quit
*****?*****

```

Enter your choice: 1

YOUR BALANCE IN Rs : 1000

```

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):
*****Welcome to ATM Service*****
1. Check Balance
2. Withdraw Cash
3. Deposit Cash

```

4. Quit

*****?*****?*****?*

Enter your choice: 2

ENTER THE AMOUNT TO WITHDRAW: 200

PLEASE COLLECT CASH

YOUR CURRENT BALANCE IS 800

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

*****Welcome to ATM Service*****

1. Check Balance

2. Withdraw Cash

3. Deposit Cash

4. Quit

*****?*****?*****?*

Enter your choice: 3

ENTER THE AMOUNT TO DEPOSIT 5000

YOUR BALANCE IS 5800

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

*****Welcome to ATM Service*****

1. Check Balance

2. Withdraw Cash

3. Deposit Cash

4. Quit

*****?*****?*****?*

Enter your choice: 1

YOUR BALANCE IN Rs : 5800

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

*****Welcome to ATM Service*****

1. Check Balance

2. Withdraw Cash

3. Deposit Cash

4. Quit

*****?*****?*****?*

Enter your choice: 4

THANK U USING ATM

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

*****Welcome to ATM Service*****

1. Check Balance

2. Withdraw Cash

3. Deposit Cash

4. Quit

*****?*****?*****?*

Enter your choice: 4

THANK U USING ATM

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

*****Welcome to ATM Service*****

```
1. Check Balance  
2. Withdraw Cash  
3. Deposit Cash  
4. Quit  
*****?*****?
```

Enter your choice: n

THANK U USING ATM

DO U WISH TO HAVE ANOTHER TRANSCATION?(y/n):

THANKS FOR USING OUT ATM SERVICE.

91. C Program to Reverse a Stack using Recursion

[« Prev](#)

[Next »](#)

This C program, using recursion, reverses a stack content. Stack here is represented using a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*  
2. * C Program to Reverse a Stack using Recursion  
3. */  
4. #include <stdio.h>  
5. #include <stdlib.h>  
6.  
7. struct node  
8. {  
9.     int a;  
10.    struct node *next;  
11.};
```

```

12.
13.void generate(struct node **);
14.void display(struct node *);
15.void stack_reverse(struct node **, struct node **);
16.void delete(struct node **);
17.
18.int main()
19.{
20.    struct node *head = NULL;
21.
22.    generate(&head);
23.    printf("\nThe sequence of contents in stack\n");
24.    display(head);
25.    printf("\nInversing the contents of the stack\n");
26.    if (head != NULL)
27.    {
28.        stack_reverse(&head, &(head->next));
29.    }
30.    printf("\nThe contents in stack after reversal\n");
31.    display(head);
32.    delete(&head);
33.
34.    return 0;
35.}
36.
37.void stack_reverse(struct node **head, struct node **head_next)
38.{
39.    struct node *temp;
40.
41.    if (*head_next != NULL)
42.    {
43.        temp = (*head_next)->next;
44.        (*head_next)->next = (*head);
45.        *head = *head_next;
46.        *head_next = temp;
47.        stack_reverse(head, head_next);
48.    }
49.}
50.
51.void display(struct node *head)
52.{
53.    if (head != NULL)
54.    {
55.        printf("%d ", head->a);
56.        display(head->next);
57.    }
58.}
59.
60.void generate(struct node **head)
61.{
62.    int num, i;
63.    struct node *temp;
64.
65.    printf("Enter length of list: ");
66.    scanf("%d", &num);
67.    for (i = num; i > 0; i--)
68.    {
69.        temp = (struct node *)malloc(sizeof(struct node));
70.        temp->a = i;
71.        if (*head == NULL)
72.        {
73.            *head = temp;
74.            (*head)->next = NULL;
75.        }
76.        else
77.        {

```

```
78.     temp->next = *head;
79.     *head = temp;
80. }
81. }
82.}
83.
84.void delete(struct node **head)
85.{
86.    struct node *temp;
87.    while (*head != NULL)
88.    {
89.        temp = *head;
90.        *head = (*head)>next;
91.        free(temp);
92.    }
93.}
```

advertisement

```
$ cc pgm40.c
$ a.out
Enter length of list: 10
```

The sequence of contents **in** stack

1 2 3 4 5 6 7 8 9 10

Inversing the contents of the stack

The contents **in** stack after reversal

10 9 8 7 6 5 4 3 2 1

92. C Program to Find the Sum of H.P Series

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of H.P series.

Problem Description

This C Program calculates the sum of H.P series.

Problem Solution

This program is used to find the sum of the harmonic progression series. Here H.P stands for harmonic progression. Harmonic progression is a progression formed by taking the reciprocals of an arithmetic progression.

Program/Source Code

Here is source code of the C Program to Find the the sum of H.P series. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Sum of H.P Series
 */
#include <stdio.h>

void main()
{
    int n;
    float i, sum, term;

    printf("1 + 1 / 2 + 1 / 3 +.....+1 / n \n");
    printf("Enter the value of n \n");
    scanf("%d", &n);
    sum = 0;
    for (i = 1; i <= n; i++)
    {
        term = 1 / i;
        sum = sum + term;
    }
    printf("the Sum of H.P Series is = %f", sum);
}
```

Program Explanation

In this C program, we are reading the limit to compute the harmonic progression from the series $1 + 1 / 2 + 1 / 3 +.....+1 / n$ using ‘n’ integer variable. Harmonic progression is a progression formed by taking the reciprocals of an arithmetic progression.

For loop is used to perform the addition for each integer values in the harmonic series up to the limit as mentioned by user in ‘n’ variable. Print the sum of H.P series using printf statement.

Runtime Test Cases

```
Output:
$ cc pgm23.c
$ a.out
1 + 1 / 2 + 1 / 3 +.....+1 / n
Enter the value of n
5
the Sum of H.P Series is = 2.283334
```

93. C Program to Find whether a Number is Prime or Not using Recursion

[« Prev](#)

[Next »](#)

This is a C program to find whether a number is prime or not using recursion.

Problem Description

The following C program, using recursion, finds whether the entered number is a prime number or not.

Problem Solution

A prime number is an integer that has no integral factor but itself and 1.

advertisement

Program/Source Code

Here is the source code of the C program to find an element in a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find whether a Number is Prime or Not using Recursion
 */
#include <stdio.h>

int primeno(int, int);

int main()
{
    int num, check;
    printf("Enter a number: ");
    scanf("%d", &num);
    check = primeno(num, num / 2);
    if (check == 1)
    {
        printf("%d is a prime number\n", num);
    }
    else
    {
        printf("%d is not a prime number\n", num);
    }
    return 0;
}

int primeno(int num, int i)
{
    if (i == 1)
    {
        return 1;
    }
    else
    {
        if (num % i == 0)
        {
            return 0;
        }
        else
        {
            return primeno(num, i - 1);
        }
    }
}
```

Program Explanation

In this C program, we are reading the integer number using ‘num’ variable. A prime number is an integer that has no integral factor but itself and 1. The check variable is used to call the primeno() function by passing the value of ‘num’ variable and the value of division of ‘num’ variable value by 2 as an argument.

advertisement

The primeno() function is used to find whether the entered number is a prime number or not. If else condition statement is used to check the value of ‘i’ variable is equal to 1 and return the value of ‘i’ variable to the called variable ‘check’.

Otherwise, if the condition is false execute the else statement and call the primeno() function by passing the value of ‘num’ variable and the decrement the value of ‘i’ variable by 1. Return the resulted value to the called variable ‘check’.

If else condition statement is used to check that the value of ‘check’ variable is equal to 1. If the condition is true print the statement as prime number. Otherwise, if the condition is false print the statement as not a prime number.

advertisement

Runtime Test Cases

```
$ cc pgm24.c
$ a.out
Enter a number: 456
456 is not a prime number

$ a.out
Enter a number: 89
89 is a prime number
```

94. C Program Find the Length of the Linked List without using Recursion

[« Prev](#)

[Next »](#)

This C Program, using iteration, counts the number of nodes in a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to count the number of nodes in a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program find the Length of the Linked List without using Recursion
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
```

```

9.    int a;
10.   struct node *next;
11. };
12.
13.
14.void generate(struct node **);
15.int length(struct node* );
16 void delete(struct node **);
17.
18.int main()
19.{
20.   struct node *head = NULL;
21.   int count;
22.
23.   generate(&head);
24.   count = length(head);
25.   printf("The number of nodes are: %d\n", count);
26.   delete(&head);
27.
28.   return 0;
29.}
30.
31.void generate(struct node **head)
32.{
33. /*for unknown number of nodes use num = rand() % 20; */
34. int num = 10, i;
35. struct node *temp;
36.
37. for (i = 0; i < num; i++)
38. {
39.   temp = (struct node *)malloc(sizeof(struct node));
40.   temp->a = i;
41.   if (*head == NULL)
42.   {
43.     *head = temp;
44.     (*head)->next = NULL;
45.   }
46.   else
47.   {
48.     temp->next = *head;
49.     *head = temp;
50.   }
51. }
52.}
53.
54.int length(struct node *head)
55.{
56.   int num = 0;
57.   while (head != NULL)
58.   {
59.     num += 1;
60.     head = head->next;
61.   }
62.   return num;
63.}
64.
65.void delete(struct node **head)
66.{
67.   struct node *temp;
68.   while (*head != NULL)
69.   {
70.     temp = *head;
71.     *head = (*head)->next;
72.     free(temp);
73.   }
74.}

```

```
$ gcc numbernode.c -o numbernode
```

```
$ a.out
```

```
The number of nodes are: 10
```

95. C Program to Perform Matrix Multiplication using Recursion

[« Prev](#)

[Next »](#)

The following C program, using recursion, performs Matrix multiplication of two matrices and displays the result. We use 2 D array to represent a matrix and resulting matrix is stored in a different matrix.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2.  * C Program to Perform Matrix Multiplication using Recursion
3.  */
4. #include <stdio.h>
5.
6. void multiply(int, int, int [][][10], int, int, int [][][10], int [][][10]);
7. void display(int, int, int[][][10]);
```

```

8.
9. int main()
10.{
11.    int a[10][10], b[10][10], c[10][10] = {0};
12.    int m1, n1, m2, n2, i, j, k;
13.
14.    printf("Enter rows and columns for Matrix A respectively: ");
15.    scanf("%d%d", &m1, &n1);
16.    printf("Enter rows and columns for Matrix B respectively: ");
17.    scanf("%d%d", &m2, &n2);
18.    if (n1 != m2)
19.    {
20.        printf("Matrix multiplication not possible.\n");
21.    }
22.    else
23.    {
24.        printf("Enter elements in Matrix A\n");
25.        for (i = 0; i < m1; i++)
26.            for (j = 0; j < n1; j++)
27.            {
28.                scanf("%d", &a[i][j]);
29.            }
30.        printf("\nEnter elements in Matrix B:\n");
31.        for (i = 0; i < m2; i++)
32.            for (j = 0; j < n2; j++)
33.            {
34.                scanf("%d", &b[i][j]);
35.            }
36.        multiply(m1, n1, a, m2, n2, b, c);
37.    }
38.    printf("On matrix multiplication of A and B the result is:\n");
39.    display(m1, n2, c);
40.}
41.
42.void multiply (int m1, int n1, int a[10][10], int m2, int n2, int b[10][10], int c[10][10])
43.{
44.    static int i = 0, j = 0, k = 0;
45.
46.    if (i >= m1)
47.    {
48.        return;
49.    }
50.    else if (i < m1)
51.    {
52.        if (j < n2)
53.        {
54.            if (k < n1)
55.            {
56.                c[i][j] += a[i][k] * b[k][j];
57.                k++;
58.                multiply(m1, n1, a, m2, n2, b, c);
59.            }
60.            k = 0;
61.            j++;
62.            multiply(m1, n1, a, m2, n2, b, c);
63.        }
64.        j = 0;
65.        i++;
66.        multiply(m1, n1, a, m2, n2, b, c);
67.    }
68.}
69.
70.void display(int m1, int n2, int c[10][10])
71.{
72.    int i, j;
73.

```

```
74. for (i = 0; i < m1; i++)
75. {
76.     for (j = 0; j < n2; j++)
77.     {
78.         printf("%d ", c[i][j]);
79.     }
80.     printf("\n");
81. }
82.}
```

advertisement

```
$ cc pgm23.c
$ a.out
Enter rows and columns for Matrix A respectively: 2
2
Enter rows and columns for Matrix B respectively: 2
2
Enter elements in Matrix A:
12 56
45 78

Enter elements in Matrix B:
2 6
5 8
On matrix multiplication of A and B the result is:
304 520
480 894
```

96. C Program to Search for an Element in the Linked List without using Recursion

[« Prev](#)

[Next »](#)

This C program, using iteration, searches for an element in a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to search for an element in a linked list. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Search for an Element in the Linked List without
3. * using Recursion
4. */
5.
6. #include <stdio.h>
```

```
7. #include <stdlib.h>
8.
9. struct node
10.{
11.    int a;
12.    struct node *next;
13.};
14.
15.void generate(struct node **, int);
16.void search(struct node *, int);
17.void delete(struct node **);
18.
19.int main()
20.{
21.    struct node *head = NULL;
22.    int key, num;
23.
24.    printf("Enter the number of nodes: ");
25.    scanf("%d", &num);
26.    printf("\nDisplaying the list\n");
27.    generate(&head, num);
28.    printf("\nEnter key to search: ");
29.    scanf("%d", &key);
30.    search(head, key);
31.    delete(&head);
32.
33.    return 0;
34.}
35.
36.void generate(struct node **head, int num)
37.{
38.    int i;
39.    struct node *temp;
40.
41.    for (i = 0; i < num; i++)
42.    {
43.        temp = (struct node *)malloc(sizeof(struct node));
44.        temp->a = rand() % num;
45.        if (*head == NULL)
46.        {
47.            *head = temp;
48.            temp->next = NULL;
49.        }
50.        else
51.        {
52.            temp->next = *head;
53.            *head = temp;
54.        }
55.        printf("%d ", temp->a);
56.    }
57.}
58.
59.void search(struct node *head, int key)
60.{
61.    while (head != NULL)
62.    {
63.        if (head->a == key)
64.        {
65.            printf("key found\n");
66.            return;
67.        }
68.        head = head->next;
69.    }
70.    printf("Key not found\n");
71.}
72.
```

```
73.void delete(struct node **head)
74.{
75.    struct node *temp;
76.
77.    while (*head != NULL)
78.    {
79.        temp = *head;
80.        *head = (*head)->next;
81.        free(temp);
82.    }
83.}
```

advertisement

```
$ gcc search_iter.c -o search_iter
```

```
$ a.out
```

```
Enter the number of nodes: 10
```

```
Displaying the list
```

```
3 6 7 5 3 5 6 2 9 1
```

```
Enter key to search: 2
```

```
key found
```

97. C Program Count the Occurrences of an Element in the Linked List without using Recursion

[« Prev](#)

[Next »](#)

This C Program, using iteration, finds the occurrence for an element in an unsorted list. The user enters the element need to be counted.

Here is the source code of the C program to find the number of occurrences of a given number in a list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
```

```

2. * C Program Count the Number of Occurrences of an Element in the Linked List
3. * without using Recursion
4. */
5. #include <stdio.h>
6.
7. int occur(int [], int, int);
8.
9. int main()
10. {
11.     int size, key, count;
12.     int list[20];
13.     int i;
14.
15.     printf("Enter the size of the list: ");
16.     scanf("%d", &size);
17.     printf("Printing the list:\n");
18.     for (i = 0; i < size; i++)
19.     {
20.         list[i] = rand() % size;
21.         printf("%d ", list[i]);
22.     }
23.     printf("\nEnter the key to find it's occurrence: ");
24.     scanf("%d", &key);
25.     count = occur(list, size, key);
26.     printf("%d occurs for %d times.\n", key, count);
27.     return 0;
28. }
29.
30. int occur(int list[], int size, int key)
31. {
32.     int i, count = 0;
33.
34.     for (i = 0; i < size; i++)
35.     {
36.         if (list[i] == key)
37.         {
38.             count += 1;
39.         }
40.     }
41.     return count;
42. }
```

advertisement

```
$ gcc occurnumber.c -o occurnumber
$ a.out
```

Enter the **size** of the list: 10

Printing the list:

3 6 7 5 3 5 6 2 9 1

Enter the key to **find** it's occurrence: 3

3 occurs for 2 times.

98. C Program to Find LCM of a Number using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to find lcm of a number using recursion.

Problem Description

This C Program Finds LCM of a Number using Recursion.

Problem Solution

The following C program, using recursion, finds the LCM. An LCM is the lowest common multiple of any 2 numbers.

advertisement

Program/Source Code

Here is the source code of the C program to find LCM of a Number using Recursion. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find LCM of a Number using Recursion
 */
#include <stdio.h>

int lcm(int, int);

int main()
{
    int a, b, result;
    int prime[100];

    printf("Enter two numbers: ");
    scanf("%d%d", &a, &b);
    result = lcm(a, b);
    printf("The LCM of %d and %d is %d\n", a, b, result);
    return 0;
}

int lcm(int a, int b)
{
    static int common = 1;

    if (common % a == 0 && common % b == 0)
    {
        return common;
    }
    common++;
    lcm(a, b);
    return common;
}
```

Program Explanation

In this C program, we are reading the two integer numbers using ‘a’ and ‘b’ variables respectively. The lcm() function is used to find LCM of a number using recursion.

advertisement

Assign the value of ‘common’ variable as 1. If condition statement is used to check the modulus of the value of ‘common’ variable by the value of ‘a’ variable is equal to 0. Also the modulus of the value of ‘common’ variable by the value of ‘b’ variable is equal to 0 using AND operation. If the condition is true, then execute the statement and return the ‘common’ value. Print the LCM of a number using the printf statement.

Runtime Test Cases

```
$ cc pgm22.c
$ a.out
Enter two numbers: 456
```

12

The LCM of 456 and 12 is 456

\$ a.out

Enter two numbers: 45 75

The LCM of 45 and 75 is 225

99. C Program Count the Number of Occurrences of an Element in the Linked List using Recursion

[« Prev](#)

[Next »](#)

This C Program uses recursive function & finds the occurrence for an element in an unsorted list. The user enters the element need to be counted.

Here is the source code of the C program to find the number of occurrences of a given number in a list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program to find the number of occurrences of a given number in a
```

```

3.  * list
4.  */
5. #include <stdio.h>
6.
7. void occur(int [], int, int, int, int *);
8.
9. int main()
10. {
11.     int size, key, count = 0;
12.     int list[20];
13.     int i;
14.
15.     printf("Enter the size of the list: ");
16.     scanf("%d", &size);
17.     printf("Printing the list:\n");
18.     for (i = 0; i < size; i++)
19.     {
20.         list[i] = rand() % size;
21.         printf("%d ", list[i]);
22.     }
23.     printf("\nEnter the key to find it's occurrence: ");
24.     scanf("%d", &key);
25.     occur(list, size, 0, key, &count);
26.     printf("%d occurs for %d times.\n", key, count);
27.     return 0;
28. }
29.
30. void occur(int list[], int size, int index, int key, int *count)
31. {
32.     if (size == index)
33.     {
34.         return;
35.     }
36.     if (list[index] == key)
37.     {
38.         *count += 1;
39.     }
40.     occur(list, size, index + 1, key, count);
41. }
```

advertisement

```

$ cc pgm13.c
$ a.out
Enter the size of the list: 7
Printing the list:
1 4 2 5 1 3 3
Enter the key to find it's occurrence: 3
3 occurs for 2 times.
```

100. C Program to Display the Nodes of a Linked List in Reverse without using Recursion

[« Prev](#)

[Next »](#)

This C program, using iteration, displays a linked list in reverse. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2.  * C Program to Display the Nodes of a Linked List in Reverse without
3.  * using Recursion
4.  */
5.
```

```
6. #include <stdio.h>
7. #include <stdlib.h>
8.
9. struct node
10. {
11.     int visited;
12.     int a;
13.     struct node *next;
14. };
15.
16. void generate(struct node **);
17. void display(struct node *);
18. void linear(struct node *);
19. void delete(struct node **);
20.
21. int main()
22. {
23.     struct node *head = NULL;
24.
25.     generate(&head);
26.     printf("\nPrinting the list in linear order\n");
27.     linear(head);
28.     printf("\nPrinting the list in reverse order\n");
29.     display(head);
30.     delete(&head);
31.
32.     return 0;
33. }
34.
35. void display(struct node *head)
36. {
37.     struct node *temp = head, *prev = head;
38.
39.     while (temp->visited == 0)
40.     {
41.         while (temp->next != NULL && temp->next->visited == 0)
42.         {
43.             temp = temp->next;
44.         }
45.         printf("%d ", temp->a);
46.         temp->visited = 1;
47.         temp = head;
48.     }
49. }
50.
51. void linear(struct node *head)
52. {
53.     while (head != NULL)
54.     {
55.         printf("%d ", head->a);
56.         head = head->next;
57.     }
58.     printf("\n");
59. }
60.
61. void generate(struct node **head)
62. {
63.     int num, i;
64.     struct node *temp;
65.
66.     printf("Enter length of list: ");
67.     scanf("%d", &num);
68.     for (i = num; i > 0; i--)
69.     {
70.         temp = (struct node *)malloc(sizeof(struct node));
71.         temp->a = i;
```

```

72.     temp->visited = 0;
73.     if (*head == NULL)
74.     {
75.         *head = temp;
76.         (*head)->next = NULL;
77.     }
78.     else
79.     {
80.         temp->next = *head;
81.         *head = temp;
82.     }
83. }
84.
85.
86.void delete(struct node **head)
87.{
88.    struct node *temp;
89.    while (*head != NULL)
90.    {
91.        temp = *head;
92.        *head = (*head)->next;
93.        free(temp);
94.    }
95.}

```

advertisement

```

$ gcc revnode_iter.c -o revnode_iter
$ a.out
Enter length of list: 5

```

```

Printing the list in linear order
1 2 3 4 5

```

```

Printing the list in reverse order
5 4 3 2 1

```

101. C Program to Find Area of Parallelogram

[« Prev](#)

[Next »](#)

This is a C Program to Find Area of Parallelogram.

Problem Description

This C Program calculates the area of Parallelogram.

Problem Solution

The formula used in this program are Area = b * a where b is the length of any base, a is the corresponding altitude.

advertisement

Program/Source Code

Here is source code of the C Program to Find the area of Parallelogram. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Area of Parallelogram
 */
#include <stdio.h>

int main()
{
    float base, altitude;
    float area;

    printf("Enter base and altitude of the given Parallelogram: \n ");
    scanf("%f%f", &base, &altitude);
    area = base * altitude;
    printf("Area of Parallelogram is: %.3f\n", area);
    return 0;
}
```

Program Explanation

In this C program, library function defined in <math.h> header file is used to compute mathematical functions. We are reading the ‘base’ and ‘altitude’ of a parallelogram. To find the surface area, the following formulas is used.

advertisement

Area = base * altitude

Runtime Test Cases

Output:

```
$ cc pgm27.c
$ a.out
```

Enter base and altitude of the given Parallelogram:

```
17 19
```

Area of Parallelogram is: 323.000

102. C Program using Recursion to Search an Element in Array

[« Prev](#)

[Next »](#)

This C Program uses recursive function & searches for an element in an unsorted list and display it's position of occurrence. The user enters the element needed to be searched.

Here is the source code of the C program to search for an element in an unsorted list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to search for an element in a list using
3.  */
4. #include <stdio.h>
```

```

5.
6. int search(int [], int, int);
7. int main()
8. {
9.     int size, index, key;
10.    int list[20];
11.    int count = 0;
12.    int i;
13.
14.    printf("Enter the size of the list: ");
15.    scanf("%d", &size);
16.    index = size;
17.    printf("Printing the list:\n");
18.    for (i = 0; i < size; i++)
19.    {
20.        list[i] = rand() % size;
21.        printf("%d\t", list[i]);
22.    }
23.    printf("\nEnter the key to search: ");
24.    scanf("%d", &key);
25.    while (index > 0)
26.    {
27.        index = search(list, index - 1, key);
28.        /* In an array first position is indexed by 0 */
29.        printf("Key found at position: %d\n", index + 1);
30.        count++;
31.    }
32.    if (!count)
33.        printf("Key not found.\n");
34.    return 0;
35.}
36.int search(int array[], int size, int key)
37.{
38.    int location;
39.    if (array[size] == key)
40.    {
41.        return size;
42.    }
43.    else if (size == -1)
44.    {
45.        return -1;
46.    }
47.    else
48.    {
49.        return (location = search(array, size - 1, key));
50.    }
51.}

```

advertisement

```

$ cc search
$ a.out
Enter the size of the list: 10

Printing the list:
3       6       7       5       3       5       6       2       9       1

Enter the key to search: 5

Key found at position: 6

Key found at position: 4

```

103. C Program to Print Armstrong Number from 1 to 1000

[« Prev](#)

[Next »](#)

This is a C Program to print armstrong number from 1 to 1000.

Problem Description

This C Program print armstrong number from 1 to 1000.

Problem Solution

An Armstrong number is an n-digit base b number such that the sum of its (base b) digits raised to the power n is the number itself. Hence 153 because $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

advertisement

Program/Source Code

Here is source code of the C Program to print armstrong number from 1 to 1000.

The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Print Armstrong Number from 1 to 1000
 */
#include <stdio.h>

main()
{
    int number, temp, digit1, digit2, digit3;

    printf("Print all Armstrong numbers between 1 and 1000:\n");
    number = 001;
    while (number <= 900)
    {
        digit1 = number - ((number / 10) * 10);
        digit2 = (number / 10) - ((number / 100) * 10);
        digit3 = (number / 100) - ((number / 1000) * 10);
        temp = (digit1 * digit1 * digit1) + (digit2 * digit2 * digit2) + (digit3 * digit3 * digit3);
        if (temp == number)
        {
            printf("\n Armstrong no is:%d", temp);
        }
        number++;
    }
}
```

Program Explanation

In this C program, we are printing the Armstrong number from 1 to 1000. An Armstrong number is an n-digit base b number such that the sum of its (base b) digits raised to the power n is the number itself. Hence, 153 because $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$. We are initializing the number variable value as 001.

advertisement

Using while loop checks the condition that the value of ‘number’ variable is less than or equal to 900. If the condition is true then, execute the iteration of the loop. The ‘digit1’ variable is used to compute the division of the value of ‘number’ variable by 10 and multiply the resulting value by 10 and subtract the multiplied value with the value of ‘number’ variable.

The ‘digit2’ variable is used to compute the division of the value of ‘number’ variable by 100, multiply the resulting value with 10, divide the value of ‘number’ variable by 10 and subtract the multiplied value from the divided value.

The ‘digit3’ variable is used to compute the division of the value of ‘number’ variable by 1000, then multiply the value divided by 100 with 10 and subtract both the values. The ‘temp’ variable is used to multiply the value of digit1, digit2, digit3 variables to the power of 3 respectively. Compute the summation of all the three multiplied values. If condition statement is used to check the value of ‘temp’ and ‘number’ variables are equal, if the condition is true print the Armstrong number.

advertisement

Runtime Test Cases

Output:
\$ cc pgm44.c
\$ a.out
Print all Armstrong numbers between 1 and 1000:

```
Amstrong no is:1  
Amstrong no is:153  
Amstrong no is:370  
Amstrong no is:371  
Amstrong no is:407
```

104. C Program to Solve the Magic Squares Puzzle without using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to solve the magic squares puzzle without using recursion.

Problem Description

The following C program, using iteration, finds the magic square for a given odd sized number.

Problem Solution

A magic square is an arrangement of numbers from 1 to n^2 in an $[n \times n]$ matrix, with each number occurring exactly once, and such that the sum of the entries of any row, any column, or any main diagonal is the same.

advertisement

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Solve the Magic Squares Puzzle without using
 * Recursion
 */
#include <stdio.h>

void magicsq(int, int []);

int main()
{
    int size;
    int a[10][10];

    printf("Enter the size: ");
    scanf("%d", &size);
    if (size % 2 == 0)
    {
        printf("Magic square works for an odd numbered size\n");
    }
    else
    {
        magicsq(size, a);
    }
    return 0;
}

void magicsq(int size, int a[][10])
{
    int sqr = size * size;
    int i = 0, j = size / 2, k;

    for (k = 1; k <= sqr; ++k)
    {
        a[i][j] = k;
        i--;
        j++;

        if (k % size == 0)
        {
            i += 2;
            -j;
        }
        else
        {
            if (j == size)
            {
                j -= size;
            }
            else if (i < 0)
            {
                i += size;
            }
        }
    }
}
```

```

for (i = 0; i < size; i++)
{
    for (j = 0; j < size; j++)
    {
        printf("%d ", a[i][j]);
    }
    printf("\n");
}
printf("\n");
}

```

Program Explanation

In this C Program, we are reading the size of an array using ‘size’ variable. If condition statement is used to check whether the size is odd numbered size or even numbered size. If the size is even numbered then magic square will not work for an even numbered and exit the program.

advertisement

Otherwise, if the condition is false, then enter the size is odd numbered size, hence magic square works for an odd numbered size. Execute the else statement. The `magicsq()` function is used to find the magic square for a given odd sized number.

Using for loop arrange the numbers from 1 to n^2 in an $[n \times n]$ matrix. If else condition statement is used to check that each number is occurring exactly once. Hence the sum of the entries of any row, any column, or any main diagonal is the same. Using for loop print the magic squares puzzle.

Runtime Test Cases

```

$ cc pgm27.c
$ a.out
Enter the size: 6
Magic square works for an odd numbered size

```

```

$ a.out
Enter the size: 5
17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9

```

105. C Program Find the Length of the Linked List using Recursion

[« Prev](#)

[Next »](#)

This C Program uses recursive function & calculates the length of a string. The user enters a string to find it's length.

Here is the source code of the C program to find the length of a string. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C program to find the length of a string
3. */
4. #include <stdio.h>
5.
6. int length(char [], int);

```

```

7. int main()
8. {
9.     char word[20];
10.    int count;
11.
12.    printf("Enter a word to count it's length: ");
13.    scanf("%s", word);
14.    count = length(word, 0);
15.    printf("The number of characters in %s are %d.\n", word, count);
16.    return 0;
17. }
18.
19.int length(char word[], int index)
20.{
21.    if (word[index] == '\0')
22.    {
23.        return 0;
24.    }
25.    return (1 + length(word, index + 1));
26.}

```

advertisement

```

$ cc pgm17.c
$ a.out
Enter a word to count it's length: 5
The number of characters in 5 are 1.

$ a.out
Enter a word to count it's length: sanfoundry
The number of characters in sanfoundry are 10.

```

106. C Program to Convert Binary Code of a Number into its Equivalent Gray's Code without using Recursion

[« Prev](#)

[Next »](#)

This is a C program to convert binary code of a number into its equivalent gray's code without using recursion.

Problem Description

This C program, using iteration, evaluates the gray code equivalent of a binary number.

Problem Solution

A gray is also represented using 0s and 1s. The specialty of gray code is that only one bit is changed in 2 consecutive numbers, say 3 and 4.

advertisement

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Convert Binary Code of a Number into its Equivalent
 * Gray's Code without using Recursion
 */
#include <stdio.h>
#include <math.h>

int bintogray(int);

int main ()
{
    int bin, gray;

    printf("Enter a binary number: ");
    scanf("%d", &bin);
    gray = bintogray(bin);
    printf("The gray code of %d is %d\n", bin, gray);
    return 0;
}

int bintogray(int bin)
{
    int a, b, result = 0, i = 0;

    while (bin != 0)
    {
        a = bin % 10;
        bin = bin / 10;
        b = bin % 10;
        if((a && !b) || (!a && b))
        {
            result = result + pow(10, i);
        }
        i++;
    }
    return result;
}
```

Program Explanation

In this C program, we are reading a binary number using ‘bin’ variable, A gray is also represented using 0s and 1s. The specialty of gray code is that only one bit is changed in 2 consecutive numbers, say 3 and 4.

advertisement

The bintogray() function is used to evaluate the gray code equivalent of a binary number by passing the value of ‘bin’ variable as argument. While loop is used to check the value of ‘bin’ variable is not equal to 0. If the condition is true execute the loop. Compute the modulus of the value of ‘bin’ variable by 10.

Divide the value of ‘num’ variable by 10. Compute the modulus of the value of ‘bin’ variable value by 10.

If condition statement is used to check the value of ‘a’ variable and the negation of the value of ‘b’ variable is true or the negation of the value ‘a’ variable and the value of ‘b’ variable is true by using the logical OR operator.

advertisement

If the condition is true execute the statement compute the value of 10 to the power of the value of ‘i’ variable and assign the value to ‘result’ variable. Print the binary code of a number into its equivalent gray’s code using printf statement.

Runtime Test Cases

```
$ cc pgm26.c -lm  
$ a.out  
Enter a binary number: 1111001010  
The gray code of 1111001010 is 1000101111
```

107. C Program to Copy One String to Another using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to copy one string to another using recursion.

Problem Description

This C Program Copies One String to Another using Recursion.

Problem Solution

This C Program uses recursive function & copies a string entered by user from one character array to another character array.

advertisement

Program/Source Code

Here is the source code of the C program to copy string using recursion. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Copy One String to Another using Recursion
 */
#include <stdio.h>

void copy(char [], char [], int);

int main()
{
    char str1[20], str2[20];

    printf("Enter string to copy: ");
    scanf("%[^\\n]s", str1);
    copy(str1, str2, 0);
    printf("Copying success.\n");
    printf("The first string is: %s\n", str1);
    printf("The second string is: %s\n", str2);
    return 0;
}

void copy(char str1[], char str2[], int index)
{
    str2[index] = str1[index];
    //printf ("INDEX IS %d\n", index);
    if (str1[index] == '\\0')
        return;
    copy(str1, str2, index + 1);
}
```

Program Explanation

1. Define the function prototype by using void copy(char [], char [], int);. The copy() function is used to copy one string to another using recursion.
2. Here, we are reading two string values using the ‘str1’ and ‘str2’ variables. str1 is used to store the input string, str2 is used to recursively copy the string and calls the function copy(str1, str2, 0);
3. In this C Program, the character from str1 is being copied to str2 until null is encountered in the string. index is incremented by 1 to move the recursion call to next state.
4. While child function returning the control to parent function the character in str1 is copied to str2.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered string is having only plain text.

```
$ gcc recursive-copy.c -o recursive-copy
$ ./recursive-copy
```

```
Enter string to copy: Welcome to Sanfoundry
Copying success.
```

```
The first string is: Welcome to Sanfoundry
The second string is: Welcome to Sanfoundry
```

Test case 2 – Here, the entered string is having text and numbers.

advertisement

```
$ gcc recursive-copy.c -o recursive-copy  
$ ./recursive-copy
```

Enter string to copy: Taj Mahal 011
Copying success.

The first string is: Taj Mahal 011
The second string is: Taj Mahal 011
Test case 3 – Here, the entered string is combination of text, numbers and symbols.

```
$ gcc recursive-copy.c -o recursive-copy  
$ ./recursive-copy
```

Enter string to copy: Welcome to Sanfoundry
Copying success.

The first string is: Welcome to Sanfoundry @ 123!
The second string is: Welcome to Sanfoundry @ 123!

108. C Program to Print the Alternate Elements in an Array

[« Prev](#)

[Next »](#)

This program prints the alternate elements in an array.

Problem Description

This is a C program which implements an array and prints the alternate elements of that array.

Problem Solution

1. Create an array and define its elements according to the size.
2. Using for loop, access the elements of the array, but instead of incrementing the iterator by 1, increase it by 2, so as to get alternate indexes of the array.

Program/Source Code

Here is source code of the C Program to print the alternate elements in an array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.
2. /*
3.  * C Program to Print the Alternate Elements in an Array
4.  */
5.
6. #include <stdio.h>
7. void main()
8. {
9.
10.    int array[10];
11.    int i;
12.    printf("enter the element of an array \n");
13.    for (i = 0; i < 10; i++)
14.        scanf("%d", &array[i]);
15.
16.    printf("Alternate elements of a given array \n");
17.    for (i = 0; i < 10; i += 2)
18.        printf( "%d\n", array[i]);
19. }
```

Program Explanation

1. Declare an array of some fixed capacity, 10.
2. Define all its elements using scanf() function under for loop.
3. Now, run a for loop with an iterator i, incrementing it by 2 each time so as to get alternate alternate indexes.
4. Alternate indexes will give alternate elements of the array.
5. Print the alternative elements as well.

advertisement

Runtime Test Cases

```
12
23
45
57
68
73
84
97
120
125
Alternate elements of a given array
12
45
68
84
120
```

109. C Program to Solve Tower-of-Hanoi Problem using Recursion

« [Prev](#)

[Next](#) »

This C Program uses recursive function & solves the tower of hanoi. The tower of hanoi is a mathematical puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top. We have to obtain the same stack on the third rod.

Here is the source code of the C program for solving towers of hanoi. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C program for Tower of Hanoi using Recursion
3.  */
4. #include <stdio.h>
5.
6. void towers(int, char, char, char);
7.
8. int main()
9. {
10.    int num;
```

```

11.
12. printf("Enter the number of disks : ");
13. scanf("%d", &num);
14. printf("The sequence of moves involved in the Tower of Hanoi are :\n");
15. towers(num, 'A', 'C', 'B');
16. return 0;
17.}
18.void towers(int num, char frompeg, char topeg, char auxpeg)
19.{ 
20. if (num == 1)
21. {
22.     printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
23.     return;
24. }
25. towers(num - 1, frompeg, auxpeg, topeg);
26. printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
27. towers(num - 1, auxpeg, topeg, frompeg);
28.}

```

advertisement

\$ **cc** pgm11.c

\$ a.out

Enter the number of disks : 3

The sequence of moves involved in the Tower of Hanoi are :

Move disk 1 from peg A to peg C
 Move disk 2 from peg A to peg B
 Move disk 1 from peg C to peg B
 Move disk 3 from peg A to peg C
 Move disk 1 from peg B to peg A
 Move disk 2 from peg B to peg C
 Move disk 1 from peg A to peg C

110. C Program to Find the First Capital Letter in a String without using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to find the first capital letter in a string without using recursion.

Problem Description

The following C program, using iteration, finds the first capital letter that exists in a string.

Problem Solution

We have included ctype.h in order to make use of “int isupper(char);” function that’s defined inside the ctype.h header file. The isupper function returns 1 if the passed character is an uppercase and returns 0 if the passed character is lowercase.

advertisement

Program/Source Code

Here is the source code of the C program to find the first capital letter in a string using recursion. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find the First Capital Letter in a String without
 * using Recursion
 */
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char caps_check(char *);

int main()
{
    char string[20], letter;

    printf("Enter a string to find it's first capital letter: ");
    scanf("%s", string);
    letter = caps_check(string);
    if (letter == 0)
    {
        printf("No capital letter is present in %s.\n", string);
    }
    else
    {
        printf("The first capital letter in %s is %c.\n", string, letter);
        return 0;
    }
}

char caps_check(char *string)
{
    int i = 0;
    while (string[i] != '\0')
    {
        if (isupper(string[i]))
        {
            return string[i];
        }
        i++;
    }
    return 0;
}
```

Program Explanation

In this C program, library function defined in < ctype.h > header file is used to compute mathematical functions. We are reading a string using string[] array variable.

advertisement

The caps_check() function is used to find the first capital letter in a string. While loop is used to check till the end of the character is present in a string is the first capital letter in that string.

If condition statement is used to check the character present in a string is an uppercase using isupper(). If the condition is true then it will return 1. Otherwise, if the condition is false then execute else statement and return the value as 0.

If else condition statement check the value of ‘length’ variable is equal to 0. If the condition is true print the output as no capital letter is present in the string. Otherwise, if the condition is false then execute the else condition statement and print the output as the first capital letter in the given string.

advertisement

Runtime Test Cases

```
$ cc pgm35.c
$ a.out
Enter a string to find it's first capital letter: prOgraMmInG
The first capital letter in prOgraMmInG is O.
```

111. C Program to Increment every Element of the Array by one & Print Incremented Array

[« Prev](#)

[Next »](#)

This program increments every element of the array by one & print incremented array.

Problem Description

This is a C program which increments every element of the array by one & print array. We need to add 1 to all the elements of the given array.

Problem Solution

1. Create an array of some size and define its elements.
2. Create a function in which the array created will be passed as parameter.
3. Inside this function, using for loop, access each element of the array, add 1 to the element and store this new value in the same place.
4. Print the array.

advertisement

Program/Source Code

Here is source code of the C Program to increments every element of the array by one & print array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to Increment every Element of the Array by one & Print Incremented Array
3.  */
4.
5.
6. #include <stdio.h>
7. void incrementArray(int[]);
8. void main()
9. {
10.
11.     int i;
12.     int array[4] = {10, 20, 30, 40};
13.     incrementArray(array);
14.     for (i = 0; i < 4; i++)
15.         printf("%d\n", array[i]); // Prints 2, 3, 4, 5
16.
17. }
18.
19. void incrementArray(int arr[])
20. {
21.
22.     int i;
23.     for (i = 0; i < 4; i++)
24.         arr[i]++; // this alters values in array in main()
25.
26. }
```

Program Explanation

1. Declare an array of some fixed size(i.e 4) and define all its element at the time of declaration.
2. Create a function an pass this array to this function as a parameter.
3. Inside this for loop, run a for loop from 0 to size-1, accessing each element of the array, add 1 to it and store the result in the same array index.
4. Print the array.

advertisement

Runtime Test Cases

11 21 31 41

112. C Program to Print Binary Equivalent of an Integer using Recursion

[« Prev](#)

[Next »](#)

This is a C program to print binary equivalent of an integer using recursion.

Problem Description

This C program, using recursion, finds the binary equivalent of a decimal number entered by the user.

Problem Solution

Decimal numbers are of base 10 while binary numbers are of base 2.

advertisement

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Print Binary Equivalent of an Integer using Recursion
```

```

/*
#include <stdio.h>

int binary_conversion(int);

int main()
{
    int num, bin;

    printf("Enter a decimal number: ");
    scanf("%d", &num);
    bin = binary_conversion(num);
    printf("The binary equivalent of %d is %d\n", num, bin);
}

int binary_conversion(int num)
{
    if (num == 0)
    {
        return 0;
    }
    else
    {
        return (num % 2) + 10 * binary_conversion(num / 2);
    }
}

```

Program Explanation

In this C program, we are reading a decimal number using ‘num’ variable. Decimal numbers are of base 10, while binary numbers are of base 2. The `binary_conversion()` function is used to find the binary equivalent of a decimal number entered by the user.

advertisement

In `binary_conversion()` function, convert the binary number to its equivalent decimal value. If `else` condition statement is used to check the value of ‘num’ variable is equal to 0. If the condition is true, execute the statement by returning 0 to the called variable ‘bin’.

Otherwise if the condition is false, execute `else` statement. Compute the modulus of the value of ‘num’ variable by 2 and add the resulted value to 10. Multiply the resulted value with the value of `binary_conversion()` function. Divide the value of ‘num’ variable by 2 and pass as an argument and execute the function recursively. Print the Binary equivalent of an integer using recursion.

Runtime Test Cases

```

$ gcc binary_recr.c -o binary_recr
$ a.out
Enter a decimal number: 10
The binary equivalent of 10 is 1010

```

113. C Program to Append the Content of File at the end of Another

[« Prev](#)

[Next »](#)

This C Program appends the content of file at the end of another.

Here is source code of the C Program to append the content of file at the end of another. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Append the Content of File at the end of Another
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. main()
8. {
9.     FILE *fsring1, *fsring2, *ftemp;
10.    char ch, file1[20], file2[20], file3[20];
11.
12.    printf("Enter name of first file ");
13.    gets(file1);
14.    printf("Enter name of second file ");
15.    gets(file2);
16.    printf("Enter name to store merged file ");
17.    gets(file3);
18.    fsring1 = fopen(file1, "r");
19.    fsring2 = fopen(file2, "r");
20.    if (fsring1 == NULL || fsring2 == NULL)
```

```

21. {
22.     perror("Error has occurred");
23.     printf("Press any key to exit...\n");
24.     exit(EXIT_FAILURE);
25. }
26. ftemp = fopen(file3, "w");
27. if (ftemp == NULL)
28. {
29.     perror("Error has occurred");
30.     printf("Press any key to exit...\n");
31.     exit(EXIT_FAILURE);
32. }
33. while ((ch = fgetc(fstring1)) != EOF)
34.     fputc(ch, ftemp);
35. while ((ch = fgetc(fstring2)) != EOF)
36.     fputc(ch, ftemp);
37. printf("Two files merged %s successfully.\n", file3);
38. fclose(fstring1);
39. fclose(fstring2);
40. fclose(ftemp);
41. return 0;
42.

```

advertisement

```

$ cc pgm47.c
$ a.out
Enter name of first file a.txt
Enter name of second file b.txt
Enter name to store merged file merge.txt
Two files merged merge.txt successfully.

```

114. C Program to Find the Length of the String using Recursion

[« Prev](#)

[Next »](#)

This C Program uses recursive function & counts the number of nodes in a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to count the number of nodes in a linked list. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * Recursive C program to find length of a linked list
3. */
4. #include <stdio.h>
5.
6. int find_len (char [], int);
7.
8. int main ()
9. {
10. char str[100];
11. int len = 0;
12.
13. printf ("Enter the string: \n");
14. scanf ("%[^\\n]s", str);
15.
16. len = find_len (str, 0);
17.

```

```

18. printf ("The length of the given string is: %d\n", len);
19. return 0;
20.}
21.
22.int find_len (char str[], int index)
23.{
24. static int l = 0;
25.
26. if (str[index] == '\0')
27.     return l;
28. else
29.     l++;
30.
31. find_len (str, index + 1);
32.}

```

advertisement

Enter the string:

Sanfoundry C Programming

The length of the given string is: 24

Enter the string:

Programming Examples

The length of the given string is: 20

115. C Program to Print the Alternate Nodes in a Linked List without using Recursion

[« Prev](#)

[Next »](#)

This C program, using iteration, displays the alternate nodes in a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Print the Alternate Nodes in a Linked List without
3. * using Recursion
4. */
5. #include <stdio.h>
6. #include <stdlib.h>
7.
8. struct node
9. {
10.     int a;
11.     struct node *next;
12. };
13.
14. void generate(struct node **);
15. void display(struct node *);
16. void delete(struct node **);
17.
18. int main()

```

```

19.
20. struct node *head = NULL;
21.
22. generate(&head);
23. printf("\nDisplaying the alternate nodes\n");
24. display(head);
25. delete(&head);
26.
27. return 0;
28.
29.
30.void display(struct node *head)
31.{
32. int flag = 0;
33.
34. while(head != NULL)
35. {
36.     if(!(flag % 2))
37.     {
38.         printf("%d ", head->a);
39.     }
40.     flag++;
41.     head = head->next;
42. }
43.
44.
45.void generate(struct node **head)
46.{
47. int num, i;
48. struct node *temp;
49.
50. printf("Enter length of list: ");
51. scanf("%d", &num);
52. for (i = num; i > 0; i--)
53. {
54.     temp = (struct node *)malloc(sizeof(struct node));
55.     temp->a = i;
56.     if (*head == NULL)
57.     {
58.         *head = temp;
59.         (*head)->next = NULL;
60.     }
61.     else
62.     {
63.         temp->next = *head;
64.         *head = temp;
65.     }
66. }
67.
68.
69.void delete(struct node **head)
70.{
71. struct node *temp;
72. while (*head != NULL)
73. {
74.     temp = *head;
75.     *head = (*head)->next;
76.     free(temp);
77. }
78.}

```

advertisement

```

$ gcc alter_iter.c -o alter_iter
$ a.out
Enter length of list: 20

```

Displaying the alternate nodes
1 3 5 7 9 11 13 15 17 19

116. C Program to Convert Octal to Decimal

[« Prev](#)

[Next »](#)

This is a C program to convert octal number to decimal.

Problem Description

This program takes a octal number as input and converts it into decimal number.

Problem Solution

1. Take a octal number as input.
2. Multiply each digits of the octal number starting from the last with the powers of 8 respectively.
3. Add all the multiplied digits.
4. The total sum gives the decimal number.

advertisement

Program/Source Code

Here is source code of the C program to Convert Octal to Decimal. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Convert Octal to Decimal
3. */
4. #include <stdio.h>
5. #include <math.h>
6.
7. int main()
8. {
9.
10.    long int octal, decimal = 0;
11.    int i = 0;
12.
13.    printf("Enter any octal number: ");
14.    scanf("%ld", &octal);
15.    while (octal != 0)
16.    {
17.        decimal = decimal + (octal % 10) * pow(8, i++);
18.        octal = octal / 10;
19.    }
20.    printf("Equivalent decimal value: %ld", decimal);
21.    return 0;
22.}
```

Program Explanation

1. Take the octal number as input and store it in the variable octal.
2. Initialize the variables decimal and i to zero.
3. Obtain the remainder and quotient of the octal number. Multiply the remainder by powers of 8 using function `pow(8, i++)`, add this value to the variable decimal and store it in the variable decimal.
4. Override the variable octal with quotient.
5. Repeat the steps 3 and 4 with the quotient obtained until the quotient becomes zero.
6. Print the variable decimal as output.

advertisement

Runtime Test Cases

Output:

```
Enter any octal number: 67
Equivalent decimal value: 55
```

117. C Program to Check whether a given Number is Perfect Number

[« Prev](#)

[Next »](#)

This is a C Program to check whether a given number is perfect number.

Problem Description

This C Program checks whether a given number is perfect number.

Problem Solution

Perfect number is a number which is equal to sum of its divisor. For eg, divisors of 6 are 1,2 and 3. The sum of these divisors is 6. So 6 is called as a perfect number.

advertisement

Program/Source Code

Here is source code of the C Program to check whether a given number is perfect number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

/*
 * C Program to Check whether a given Number is Perfect Number
 */
#include <stdio.h>

int main()
{
    int number, rem, sum = 0, i;

    printf("Enter a Number\n");
    scanf("%d", &number);
    for (i = 1; i <= (number - 1); i++)
    {
        rem = number % i;
        if (rem == 0)
        {
            sum = sum + i;
        }
    }
    if (sum == number)
        printf("Entered Number is perfect number");
    else
        printf("Entered Number is not a perfect number");
    return 0;
}

```

Program Explanation

In this C program, we are reading the integer value using ‘number’ variable. Perfect number is a number which is equal to sum of its divisor. For example, divisors of 6 are 1, 2 and 3. The sum of these divisors is 6. So the number 6 is called as perfect number.

advertisement

For loop statement is used to assign the modulus of the value of ‘number’ variable by the value of ‘i’ variable. If condition statement is used to check the value of ‘rem’ variable is equal to 0, if the condition is true to execute if condition statement and compute the summation the value of ‘sum’ variable with the value of ‘i’ variable.

Another If-else condition statement is used to check that both the value of ‘sum’ variable and the value of ‘number’ variable are equal, if the condition is true print the statement as perfect number. Otherwise, execute else condition statement and print the statement as not a perfect number.

Runtime Test Cases

```

Output:
$ cc pgm42.c
$ a.out
Enter a Number
6
Entered Number is perfect number

$ a.out
Enter a Number
100
Entered Number is not a perfect number

```

118. C Program to Find the Sum of two Binary Numbers

[« Prev](#)

[Next »](#)

This is a C program to Find the Sum of two Binary Numbers.

Problem Description

This program finds the sum of two binary numbers.

Problem Solution

1. Take two binary numbers as input.
2. Add each bits from the two binary numbers separately starting from LSB.
3. The operations may be as follows.
 - a) $(0+0)=0$,
 - b) $(1+0)=1$,
 - c) $(1+1)=0$ and 1 will be remainder.

advertisement

Program/Source Code

Here is source code of the C program to Find the Sum of two Binary Numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Find the Sum of two Binary Numbers
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.
9.     long binary1, binary2;
10.    int i = 0, remainder = 0, sum[20];
11.
12.    printf("Enter the first binary number: ");
13.    scanf("%ld", &binary1);
14.    printf("Enter the second binary number: ");
15.    scanf("%ld", &binary2);
16.    while (binary1 != 0 || binary2 != 0)
17.    {
18.        sum[i++]=(binary1 % 10 + binary2 % 10 + remainder) % 2;
19.        remainder=(binary1 % 10 + binary2 % 10 + remainder) / 2;
20.        binary1 = binary1 / 10;
21.        binary2 = binary2 / 10;
22.    }
23.    if (remainder != 0)
24.        sum[i++] = remainder;
25.        -i;
26.    printf("Sum of two binary numbers: ");
27.    while (i >= 0)
28.        printf("%d", sum[i--]);
29.    return 0;
30.}

```

Program Explanation

1. Take two binary numbers as input and store it in the variables binary1 and binary2.
2. Initialize the variables i and remainder to zero.
3. Obtain the remainders of both the binary numbers.
4. Obtain the quotients of both the binary numbers.
5. Firstly add the remainders of both binary numbers and further add the variable remainder.
6. Obtain the remainder of the result got at step 5 when divided by 2 and store it in the array sum[].
7. Obtain the quotient of the result got at step 5 when divided by 2 and override the variable remainder with this value.
8. Override the variables binary1 and binary2 with their quotient got at step 4.
9. Repeat the steps 3-8 with the new values of binary1 and binary2 until both becomes zero.
10. When it becomes zero check if any remainder exists. If it is, then copy it into the array sum.
11. Print the sum as output.

advertisement

Runtime Test Cases

Output:

```

Enter the first binary number: 100000
Enter the second binary number: 101010
Sum of two binary numbers: 1001010

```

119. C Program to Extract Last two Digits of a given Year

[« Prev](#)

[Next »](#)

This is C Program to extract last two digits of a given year.

Problem Description

This program takes any year as input and prints its last two digits.

Problem Solution

1. Take any year as input.
2. Divide the input by 100 and obtain its remainder.
3. The remainder got is the output.

advertisement

Program/Source Code

Here is source code of the C Program to extract last two digits of a given year. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Extract Last two Digits of a given Year
3. */
4. #include <stdio.h>
```

```
5.  
6. int main()  
7. {  
8.     int year, yr;  
9.  
10.    printf("Enter the year ");  
11.    scanf("%d", &year);  
12.    yr = year % 100;  
13.    printf("Last two digits of year is: %02d", yr);  
14.    return 0;  
15.}
```

Program Explanation

1. Take any year as input and store it in the variable year.
2. Divide the variable year by 100 and obtain its remainder. Store the remainder in the variable yr.
3. Print the variable yr as output.

advertisement

Runtime Test Cases

Output:
Enter the year 2012
Last two digits of year is: 12

120. C Program to Find the First Capital Letter in a String using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to find the first capital letter in a string using recursion.

Problem Description

The following C program, using recursion, finds the first capital letter that exists in a string.

Problem Solution

We have included ctype.h in order to make use of “int isupper(char);” function that’s defined inside the ctype.h headerfile. The isupper function returns 1 if the passed character is an uppercase and returns 0 if the passed character is a lowercase.

advertisement

Program/Source Code

Here is the source code of the C program to find the first capital letter in a string using recursion. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
```

```

* C Program to find the first capital letter in a string using
* Recursion
*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char caps_check(char *);

int main()
{
    char string[20], letter;

    printf("Enter a string to find it's first capital letter: ");
    scanf("%s", string);
    letter = caps_check(string);
    if (letter == 0)
    {
        printf("No capital letter is present in %s.\n", string);
    }
    else
    {
        printf("The first capital letter in %s is %c.\n", string, letter);
        return 0;
    }
}

char caps_check(char *string)
{
    static int i = 0;
    if (i < strlen(string))
    {
        if (isupper(string[i]))
        {
            return string[i];
        }
        else
        {
            i = i + 1;
            return caps_check(string);
        }
    }
    else return 0;
}

```

Program Explanation

In this C program, library function defined in < ctype.h > header file is used to compute mathematical functions. We are reading a string using string[] array variable.

advertisement

The caps_check() function is used to find the first capital letter in a string using Recursion. Nested if else condition statement is used to check the value of ‘i’ variable is less than the length of the string. If the condition is true then execute the statement.

Another if condition statement is used to check the character is an uppercase using isupper() function. If the condition is true then return the value 1. Otherwise, if the condition is false, then execute else statement and return the value 0.

If else condition statement is used to check the value of the ‘length’ variable is equal to 0. If the condition is true then execute the statement and print no capital letter is present in the string. Otherwise, if the condition is false, then execute the else condition statement and print the statement as the first capital letter in the given string.

advertisement

Runtime Test Cases

```
$ cc pgm32.c
$ a.out
Enter a string to find it's first capital letter: iloveC
The first capital letter in iloveC is C.
```

121. C Program to Convert a Number Decimal System to Binary System using Recursion

[« Prev](#)

[Next »](#)

This is a C program to convert a number decimal system to binary system using recursion.

Problem Description

The following C program using recursion finds a binary equivalent of a decimal number entered by the user.

Problem Solution

The user has to enter a decimal which has a base 10 and this program evaluates the binary equivalent of that decimal number with base 2.

advertisement

Program/Source Code

Here is the source code of the C program to find the binary equivalent of the decimal number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Convert a Number Decimal System to Binary System using Recursion
 */
#include <stdio.h>
```

```

int convert(int);
int main()
{
    int dec, bin;

    printf("Enter a decimal number: ");
    scanf("%d", &dec);
    bin = convert(dec);
    printf("The binary equivalent of %d is %d.\n", dec, bin);

    return 0;
}

int convert(int dec)
{
    if (dec == 0)
    {
        return 0;
    }
    else
    {
        return (dec % 2 + 10 * convert(dec / 2));
    }
}

```

Program Explanation

In this C program, we are reading a decimal number using the ‘dec’ variable. The function convert() is used to convert a number decimal system to binary system using recursion.

advertisement

If else condition statement is used to check the value of ‘dec’ variable is equal to 0, if the condition is true execute the statement return null value. Otherwise if the condition is false then execute the else condition statement.

Divide the value of ‘dec’ variable by 2. Multiply the result value with 10 and add the value with the modulus of the value of ‘dec’ variable by 2. Print the Binary number of the decimal system using printf statement.

Runtime Test Cases

```

$ cc pgm31.c
$ a.out
Enter a decimal number: 10
The binary equivalent of 10 is 1010.

```

122. C Program to Illustrate Pass by Value

[« Prev](#)

[Next »](#)

This is a C program to illustrate pass by value.

Problem Description

The program illustrates pass by value method.

Problem Solution

1. Take two numbers as input.
2. Using a function, swap their values. You will notice that we are passing the values in the swap() function, hence the swapped values will NOT be reflected in main() function.
3. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C Program to illustrate pass by value. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Illustrate Pass by Value.
3. */
4. #include <stdio.h>
5.
6. void swap(int a, int b)
7. {
8.     int temp;
```

```
9.     temp = a;
10.    a = b;
11.    b = temp;
12.}
13.
14.int main()
15.{
16.    int num1 = 10, num2 = 20;
17.
18.    printf("Before swapping num1 = %d num2 = %d\n", num1, num2);
19.    swap(num1, num2);
20.    printf("After swapping num1 = %d num2 = %d \n", num1, num2);
21.    return 0;
22.}
```

Program Explanation

1. Take two numbers as input and store it in the variables num1 and num2 respectively.
2. Call the function swap and pass the variables num1 and num2 as parameters to the function swap.
3. In function swap, receive the parameters through variables a and b respectively.
4. Copy the value of variable a to the variable temp. Copy the value of variable b to the variable a and copy the value of variable temp to the variable b. This will do the swapping ONLY in the swap() function, but it will NOT change the value of variables in the main() function.
5. Print the variables num1 and num2 in the main function as output and exit.

advertisement

Runtime Test Cases

```
Before swapping num1 = 10 num2 = 20
After swapping num1 = 10 num2 = 20
```

123. C Program to Illustrate Pass by Reference

[« Prev](#)

[Next »](#)

This is a C Program to illustrate pass by reference.

Problem Description

This program illustrates pass by reference.

Problem Solution

1. Pass the addresses of the variables as parameters to the function.
2. In function definition receive the parameters through pointers.
3. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C Program to illustrate pass by reference. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Illustrate Pass by Reference
3.  */
4. #include <stdio.h>
5.
6. void cube( int *x);
7.
8. int main()
```

```

9. {
10.     int num = 10;
11.
12.     cube(&num);
13.     printf("the cube of the given number is %d", num);
14.     return 0;
15. }
16.
17. void cube(int *x)
18. {
19.     *x = (*x) * (*x) * (*x);
20. }
```

Program Explanation

1. Initialize the variable num to 10.
2. Call the function cube and pass the address of variable num as parameter.
3. In the function definition, receive the parameter through pointer x and compute the cube of the variable through pointer x.
4. Print the variable num as output in the main function and exit.

advertisement

Runtime Test Cases

output:
the cube of the given number is 1000

124. C Program to Find Reverse of a Number using Recursion

[« Prev](#)

[Next »](#)

This is a C program to find reverse of a number using recursion.

Problem Description

This C program finds the reverse of a number using recursion.

Problem Solution

The following C program using recursion reverses the digits of the number and displays it on the output of the terminal. Eg: 123 becomes 321.

advertisement

Program/Source Code

Here is the source code of the C program to find the reverse of a number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

/*
 * C program to find the reverse of a number using recursion
 */
#include <stdio.h>
#include <math.h>
```

```

int rev(int, int);

int main()
{
    int num, result;
    int length = 0, temp;

    printf("Enter an integer number to reverse: ");
    scanf("%d", &num);
    temp = num;
    while (temp != 0)
    {
        length++;
        temp = temp / 10;
    }
    result = rev(num, length);
    printf("The reverse of %d is %d.\n", num, result);
    return 0;
}

int rev(int num, int len)
{
    if (len == 1)
    {
        return num;
    }
    else
    {
        return (((num % 10) * pow(10, len - 1)) + rev(num / 10, --len));
    }
}

```

Program Explanation

In this C program, we are reading the integer number using the ‘num’ variable. Assign the value of ‘num’ variable to ‘temp’ variable. While loop is used to check the condition the value of ‘temp’ variable is not equal to 0, if the condition is true execute the statement divide the value of ‘temp’ variable by 10.

advertisement

The result variable is used to call the rev() function by passing ‘num’ and ‘length’ variable value as argument. The function rev() is used to reverse the digits of the number. If else condition statement is used to check the value of ‘len’ variable is equal to 1. If the condition is true execute the statement.

Otherwise, if the condition is false execute the statement. Compute the modulus of the value of ‘num’ variable by 10 integer and multiply the resulted value with 10. Compute the power of the value of ‘len’ variable using pow() function. Add the resulted value ‘num’ variable with 10. Print the reverse of a number using recursion.

Runtime Test Cases

```

$ cc pgm34.c
$ a.out
Enter an integer number to reverse: 1234
The reverse of 1234 is 4321.

```

125. C Program to Display Function without using the Main Function

[« Prev](#)

[Next »](#)

This is a C program to display function without using the main function.

Problem Description

This program displays function without using the main function.

Problem Solution

1. Use #define macro to rename the main.
2. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C Program to display function without using the main function. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to display function without using the Main Function
3.  */
4. #include <stdio.h>
5. #define decode(s,t,u,m,p,e,d) m##s##u##t
6. #define begin decode(a,n,i,m,a,t,e)
7.
8. int begin()
9. {
```

```
10. printf(" helloworld ");
11. }
```

Program Explanation

1. Use #define macro for doing this.
2. #define decode(s,t,u,m,p,e,d) m##s##u##t.
This code says, next time when decode is encountered with 7 parameters, replace it by the concatenation of the 4th,1st,3rd and 2nd argument.
3. #define begin decode(a,n,i,m,a,t,e). In this line the 4th, 1st, 3rd and 2nd argument are m, a, i and n respectively. So the main is replaced by begin.
4. Use function named begin and print the output as " helloworld " and exit.

advertisement

Runtime Test Cases

Output:

```
helloworld
```

126. C Program to Find Sum of N Numbers using Recursion

[« Prev](#)

[Next »](#)

This is a C program to find sum of first N numbers using recursion.

Problem Description

The following C program using recursion displays the first N natural number on the terminal.

Problem Solution

The user enters the Nth number as the input, the program then calculates the sum of first N numbers using recursion and then displays the final result.

advertisement

Program/Source Code

Here is the source code of the C program to display first N numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find Sum of N Numbers using Recursion
 */
#include <stdio.h>

void display_sum(int);
```

```

int main()
{
    int num;

    printf("Enter the Nth number: ");
    scanf("%d", &num);
    display_sum(num);
    return 0;
}

void display_sum(int num)
{
    static int sum = 0;

    if (num == 0)
    {
        printf("Sum of first N numbers is %d\n", sum);
        return;
    }
    else
    {
        sum += num;
        display_sum(--num);
    }
}

```

Program Explanation

In this C program, we are reading the integer number using the ‘num’ variable. To find Sum of N Numbers using Recursion, call the display_sum() by passing the num variable value as argument.

advertisement

In function display_sum(), initialize the value of ‘sum’ variable with 0 value. Here, sum variable is defined as static so that only one copy of that object will be there upon repeated invocation of that function. If else conditional statement is used to check the value of ‘num’ variable. If the value of ‘num’ variable is non zero, we will increment the value of sum variable by num and then call display_sum() recursively by reducing the value of num variable by 1.

Once num becomes 0, we know that we have completed the recursion and we will display the final result stored in the ‘sum’ variable.

Runtime Test Cases

```

$ cc pgm33.c
$ a.out
Enter the Nth number: 3
Sum of first N numbers is 6

$ a.out
Enter the Nth number: 5
Sum of first N numbers is 15

```

127. C Program to Find HCF of a given Number using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to find hcf of a given number using recursion.

Problem Description

This C Program finds HCF of a given Number using Recursion.

Problem Solution

The following C program using recursion finds the HCF of two entered integers. The HCF stands for Highest Common Factor.

advertisement

Program/Source Code

Here is the source code of the C program to find HCF of two numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find HCF of a given Number using Recursion
 */
#include <stdio.h>
```

```

int hcf(int, int);

int main()
{
    int a, b, result;

    printf("Enter the two numbers to find their HCF: ");
    scanf("%d%d", &a, &b);
    result = hcf(a, b);
    printf("The HCF of %d and %d is %d.\n", a, b, result);
}

int hcf(int a, int b)
{
    while (a != b)
    {
        if (a > b)
        {
            return hcf(a - b, b);
        }
        else
        {
            return hcf(a, b - a);
        }
    }
    return a;
}

```

Program Explanation

In this C Program, we are reading the two integer numbers using ‘a’ and ‘b’ variables respectively. The hcf() function is used to find the HCF of two entered integers using recursion. HCF is the Highest Common Factor.

advertisement

While loop is used to check that both ‘a’ and ‘b’ variable values are not equal. If the condition is true then execute the loop. Otherwise, if the condition is false it will return the value.

If else condition statement is used to check the value of ‘a’ variable is greater than the value of ‘b’ variable. If the condition is true, then return the value. Otherwise, if the condition is false, then execute else statement and return two integer variable value. Print the HCF of a given number using printf statement.

Runtime Test Cases

```

$ cc pgm32.c
$ a.out
Enter the two numbers to find their HCF: 24 36
The HCF of 24 and 36 is 12.

```

128. C Program to Find Product of 2 Numbers without using Recursion

[« Prev](#)

[Next »](#)

This is a C program to find product of 2 numbers without using recursion.

Problem Description

This C Program Finds the Product of 2 Numbers without using Recursion.

Problem Solution

This C Program using iteration, finds the product of 2 numbers without using the multiplication operator.

advertisement

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find Product of 2 Numbers without using Recursion
 */
#include <stdio.h>

int product(int, int);

int main()
{
    int a, b, result;
```

```
printf("Enter two numbers to find their product: ");
scanf("%d%d", &a, &b);
result = product(a, b);
printf("Product of %d and %d is %d\n", a, b, result);
return 0;
}

int product(int a, int b)
{
    int temp = 0;

    while (b != 0)
    {
        temp += a;
        b--;
    }
    return temp;
}
```

Program Explanation

In this C program, we are reading the two numbers. The product() function is used to compute the product of two numbers. In the product function initially declare the value of ‘temp’ variable as 0.

advertisement

The while loop is used to check the value of ‘b’ variable is not equal to 0. If the condition is true, then execute the loop. Add the value of ‘temp’ and ‘a’ variables and decrement the value of ‘b’ variable. print the product of two numbers.

Runtime Test Cases

```
$ cc pgm19.c
$ a.out
Enter two numbers to find their product: 89 458
Product of 89 and 458 is 40762
```

129. C Program to Find Volume and Surface Area of Sphere

[« Prev](#)

[Next »](#)

This is a C Program to find volume and surface area of sphere.

Problem Description

This C Program calculates the volume and surface area of sphere.

Problem Solution

The formula used in this program are Surface_area = $4 * \pi * r^2$, Volume = $\frac{4}{3} * \pi * r^3$ where r is the radius of the sphere, $\pi = 22/7$

advertisement

Program/Source Code

Here is source code of the C Program to Find the volume and surface area of sphere.

The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Volume and Surface Area of Sphere
 */
#include <stdio.h>
#include <math.h>

int main()
{
    float radius;
```

```
float surface_area, volume;  
  
printf("Enter radius of the sphere : \n");  
scanf("%f", &radius);  
surface_area = 4 * (22/7) * radius * radius;  
volume = (4.0/3) * (22/7) * radius * radius * radius;  
printf("Surface area of sphere is: %.3f", surface_area);  
printf("\n Volume of sphere is : %.3f", volume);  
return 0;  
}
```

Program Explanation

In this C program, library function defined in `<math.h>` header file is used to compute mathematical functions. We are reading ‘radius’ of the sphere. To find the surface area and volume, the following formulas are used.

advertisement

Surface area = $4 * 3.14 * \text{radius} * \text{radius}$,

Volume = $4/3 * 3.14 * \text{radius} * \text{radius} * \text{radius}$.

Runtime Test Cases

```
Output:  
$ cc pgm30.c  
$ a.out  
Enter radius of the sphere :  
40  
Surface area of sphere is: 19200.000  
Volume of sphere is : 256000.000
```

130. C Program to Reverse the String using Recursion

[« Prev](#)

[Next »](#)

This is a C Program to reverse the string using recursion.

Problem Description

This C Program reverses the string using recursion.

Problem Solution

This C Program uses recursive function & reverses the string entered by user in the same memory location. Eg: “program” will be reversed to “margorp”

advertisement

Program/Source Code

Here is the source code of the C program to reverse a string. The C Program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Reverse the String using Recursion
 */
#include <stdio.h>
#include <string.h>

void reverse(char [], int, int);
int main()
{
    char str1[20];
    int size;

    printf("Enter a string to reverse: ");
    scanf("%s", str1);
```

```

size = strlen(str1);
reverse(str1, 0, size - 1);
printf("The string after reversing is: %s\n", str1);
return 0;
}

void reverse(char str1[], int index, int size)
{
    char temp;
    temp = str1[index];
    str1[index] = str1[size - index];
    str1[size - index] = temp;
    if (index == size / 2)
    {
        return;
    }
    reverse(str1, index + 1, size);
}

```

Program Explanation

In this C program we are reading a string using ‘str1[]’ array variable. Assign the value of the length of string using `strlen()` to size variable.

advertisement

If condition statement is used to check that both the values of ‘index’ and ‘size’ variables are equal and divide the value by 2. If the condition is true then execute the statement and return the value.

Otherwise, if the condition is false then exit the statement. Again call the `reverse()` function by passing the value of ‘str1’ variable and the summation of the value of ‘index’ variable with 1 and the value of ‘size’ variable as argument. Print the reversed string from the given string.

Runtime Test Cases

```

$ cc pgm12.c
$ a.out
Enter a string to reverse: malayalam
The string after reversing is: malayalam

$ a.out
Enter a string to reverse: cprogramming
The string after reversing is: gnimmargorp

```

131. C Program to find HCF of a given Number without using Recursion

« [Prev](#)

[Next](#) »

The following C program using iteration finds the HCF of two entered integers. The HCF stands for Highest Common Factor.

Here is the source code of the C program to find the HCF of two entered integers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to find HCF of a given Number without using Recursion
3. */
4. #include <stdio.h>
5.
6. int hcf(int, int);
7.
8. int main()
9. {
10.     int a, b, result;
11.
12.     printf("Enter the two numbers to find their HCF: ");
13.     scanf("%d%d", &a, &b);
14.     result = hcf(a, b);
15.     printf("The HCF of %d and %d is %d.\n", a, b, result);
16.
17.     return 0;
18. }
19.
20.int hcf(int a, int b)
21.{
22.    while (a != b)
23.    {
24.        if (a > b)
25.        {
26.            a = a - b;
27.        }
28.        else
29.        {
30.            b = b - a;
```

```
31. }
32. }
33. return a;
34.}
```

advertisement

```
$ cc pgm31.c
$ a.out
Enter the two numbers to find their HCF: 24 36
The HCF of 24 and 36 is 12.
```

132. C Program to Reverse a Stack without using Recursion

[« Prev](#)

[Next »](#)

This C program, using iteration, reverses a stack content. Stack here is represented using a linked list. A linked list is an ordered set of data elements, each containing a link to its successor.

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Reverse a Stack without using Recursion
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
9.     int a;
10.    struct node *next;
11.};
12.
13.void generate(struct node **);
14.void display(struct node *);
15.void stack_reverse(struct node **);
16.void delete(struct node **);
17.
18.int main()
19.{
20.    struct node *head = NULL;
21.
22.    generate(&head);
23.    printf("\nThe sequence of contents in stack\n");
24.    display(head);
25.    printf("\nInversing the contents of the stack\n");
26.    stack_reverse(&head);
27.    printf("\nThe contents in stack after reversal\n");
28.    display(head);
29.    delete(&head);
30.    return 0;
```

```

31.
32.
33.void stack_reverse(struct node **head)
34.{
35.    struct node *temp, *prev;
36.
37.    if (*head == NULL)
38.    {
39.        printf("Stack does not exist\n");
40.    }
41.    else if ((*head)->next == NULL)
42.    {
43.        printf("Single node stack reversal brings no difference\n");
44.    }
45.    else if ((*head)->next->next == NULL)
46.    {
47.        (*head)->next->next = *head;
48.        *head = (*head)->next;
49.        (*head)->next = NULL;
50.    }
51.    else
52.    {
53.        prev = *head;
54.        temp = (*head)->next;
55.        *head = (*head)->next->next;
56.        prev->next = NULL;
57.        while ((*head)->next != NULL)
58.        {
59.            temp->next = prev;
60.            prev = temp;
61.            temp = *head;
62.            *head = (*head)->next;
63.        }
64.        temp->next = prev;
65.        (*head)->next = temp;
66.    }
67.}
68.
69.void display(struct node *head)
70.{
71.    if (head != NULL)
72.    {
73.        printf("%d ", head->a);
74.        display(head->next);
75.    }
76.}
77.
78.void generate(struct node **head)
79.{
80.    int num, i;
81.    struct node *temp;
82.
83.    printf("Enter length of list: ");
84.    scanf("%d", &num);
85.    for (i = num; i > 0; i--)
86.    {
87.        temp = (struct node *)malloc(sizeof(struct node));
88.        temp->a = i;
89.        if (*head == NULL)
90.        {
91.            *head = temp;
92.            (*head)->next = NULL;
93.        }
94.        else
95.        {
96.            temp->next = *head;

```

```
97.     *head = temp;
98. }
99. }
100.}
101.
102.void delete(struct node **head)
103.{
104.    struct node *temp;
105.    while (*head != NULL)
106.    {
107.        temp = *head;
108.        *head = (*head)>next;
109.        free(temp);
110.    }
111.}
```

advertisement

```
$ cc revstack_iter.c -o revstack_iter
$ a.out
Enter length of list: 8
```

The sequence of contents **in** stack

1 2 3 4 5 6 7 8

Inversing the contents of the stack

The contents **in** stack after reversal

8 7 6 5 4 3 2 1

133. C Program to Find the two Elements such that their Sum is Closest to Zero

« [Prev](#)

[Next](#) »

This is a C Program to find the two elements such that their sum is closest to zero.

Problem Description

This program finds the two elements such that their sum is closest to zero.

Problem Solution

1. Initialize the array with some numbers.
2. Firstly add the first two elements of array and let it be the minimum sum.
3. Keeping this as default, try all the combinations in the array. Use for loops for doing this.
4. Find the minimum and print the output.

advertisement

Program/Source Code

Here is source code of the C Program to find the two elements such that their sum is closest to zero. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2. /*  
3.  * C Program to Find the two Elements such that their Sum is Closest to Zero  
4.  */  
5. # include <stdio.h>  
6. # include <stdlib.h>  
7. # include <math.h>  
8.  
9. void minabsvaluepair(int array[], int array_size)  
10.{  
11.    int count = 0;  
12.    int l, r, min_sum, sum, min_l, min_r;
```

```

13.
14. /* Array should have at least two elements*/
15. if (array_size < 2)
16. {
17.     printf("Invalid Input");
18.     return;
19. }
20.
21. /* Initialization of values */
22. min_l = 0;
23. min_r = 1;
24. min_sum = array[0] + array[1];
25. for (l = 0, l < array_size - 1, l++)
26. {
27.     for (r = l + 1, r < array_size, r++)
28.     {
29.         sum = array[l] + array[r];
30.         if (abs(min_sum) > abs(sum))
31.         {
32.             min_sum = sum;
33.             min_l = l;
34.             min_r = r;
35.         }
36.     }
37. }
38. printf(" The two elements whose sum is minimum are %d and %d", array[min_l], array[min_r]);
39.
40.
41.int main()
42.
43. int array[] = {42, 15, -25, 30, -10, 35};
44. minabsvaluepair(array, 6);
45. getchar();
46. return 0;
47.

```

Program Explanation

1. Initialize the array named array[] with the random values.
2. Call the function minabsvaluepair and pass array and its size as parameters.
3. In the function definition receive the parameters through variables array[] and array_size.
4. If the array_size is less than 2, then print the output as “Invalid Input” and exit.
5. Initialize the variables min_l and min_r with 0 and 1 respectively.
6. Initialize the variable min_sum with the sum of 1st two elements of the array.
7. Using two for loops, compare all the combinations with this default sum.
8. If the minimum one exits, then change the values of variables min_l and min_r with this new array positions and exit.
9. Print the array values with the positions min_l and min_r as output and exit.

advertisement

Runtime Test Cases

The two elements whose sum is minimum are 15 and -10

134. C Program to Find Sum of the Series 1/1! + 2/2! + 3/3! +1/N!

[« Prev](#)

[Next »](#)

This is a C Program to find sum of the series $1/1! + 2/2! + 3/3! + \dots + 1/N!$.

Problem Description

This C Program calculates the Sum of Series $1/1! + 2/2! + 3/3! + \dots + 1/N!$.

Problem Solution

Take input from the user and calculates the series as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Find the Sum of Series $1/1! + 2/2! + 3/3! + \dots + 1/N!$. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find find Sum of the Series 1/1! + 2/2! + 3/3! + .....1/N!
 */
#include <stdio.h>

double sumseries(double);

main()
{
    double number,sum;
    printf("\n Enter the value: ");
    scanf("%lf", &number);
    sum = sumseries(number);
```

```
    printf("\n Sum of the above series = %lf ", sum);
}

double sumseries(double m)
{
    double sum2 = 0, f = 1, i;
    for (i = 1; i <= m; i++)
    {
        f = f * i;
        sum2 = sum2 +(i / f);
    }
    return(sum2);
}
```

Program Explanation

In this C Program, we are reading the limit using ‘number’ integer variable. The sumseries() function is used to compute the summation of the series by passing the limit ‘number’ variable value as argument.

advertisement

For loop is used to compute the summation for each integer values in the series up to the limit as mentioned by user in ‘number’ variable. Compute the factorial for the denominator by multiplying the value of ‘f’ variable with the value of ‘i’ variable.

Compute the summation of series by dividing the value of ‘i’ variable by the value of ‘f’ variable. Add the value with the value of ‘sum2’ variable. Print the sum of the series using printf statement.

Runtime Test Cases

Output:
\$ cc pgm20.c
\$ a.out

Enter the value: 5
Sum of the above series = 2.708333

135. C Program to Find 2 Elements in the Array such that Difference between them is Largest

[« Prev](#)

[Next »](#)

This C Program checks 2 elements in the array such that difference between them is largest. This program finds maximum difference between the 2 array elements.

Here is source code of the C Program to find 2 elements in the array such that difference between them is largest.. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find 2 Elements in the Array such that Difference between them is Largest
3. */
4. #include <stdio.h>
5.
6. int maximum_difference(int array[], int arr_size)
7. {
8.     int max_diff = array[1] - array[0];
9.     int i, j;
10.    for (i = 0; i < arr_size; i++)
11.    {
12.        for (j = i + 1; j < arr_size; j++)
13.        {
14.            if (array[j] - array[i] > max_diff)
15.                max_diff = array[j] - array[i];
16.        }
17.    }
18.    return max_diff;
19.}
20.
21.int main()
22.{
23.    int array[] = {10, 15, 90, 200, 110};
24.    printf("Maximum difference is %d", maximum_difference(array, 5));
25.    getchar();
26.    return 0;
27.}
```

```
$ cc pgm97.c
$ a.out
Maximum difference is 190
```

136. C Program to Convert Binary Code of a Number into its Equivalent Gray's Code using Recursion

[« Prev](#)

[Next »](#)

This is a C program to convert binary code of a number into its equivalent gray's code using recursion.

Problem Description

This C program using recursion evaluates the gray code equivalent of a binary number.

Problem Solution

A gray is also represented using 0s and 1s. The speciality of gray code is that only one bit is changed in 2 consecutive numbers, say 3 and 4.

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Convert Binary Code of a Number into its Equivalent
 * Gray's Code using Recursion
 */
#include <stdio.h>

int bintogray(int);
int main ()
```

```

{
    int bin, gray;

    printf("Enter a binary number: ");
    scanf("%d", &bin);
    gray = bintogray(bin);
    printf("The gray code of %d is %d\n", bin, gray);
    return 0;
}

int bintogray(int bin)
{
    int a, b, result = 0, i = 0;

    if (!bin)
    {
        return 0;
    }
    else
    {
        a = bin % 10;
        bin = bin / 10;
        b = bin % 10;
        if ((a && !b) || (!a && b))
        {
            return (1 + 10 * bintogray(bin));
        }
        else
        {
            return (10 * bintogray(bin));
        }
    }
}

```

Program Explanation

In this C program, we are reading a binary number using ‘bin’ variable, A gray is also represented using 0s and 1s. The specialty of gray code is that only one bit is changed in 2 consecutive numbers, say 3 and 4.

advertisement

The bintogra() function is used to evaluate the gray code equivalent of a binary number. While loop is used to check the value of ‘bin’ variable is not equal to 0. If the condition is true execute the loop. Compute the modulus of the value of ‘bin’ variable by 10 and assign to ‘a’ variable. Divide the value of ‘num’ variable by 10 and assign the value to ‘bin’ variable. Compute the modulus of the value of ‘bin’ variable by 10 and assign the value to ‘b’ variable.

If condition statement is used to check the value of ‘a’ variable and the negation of the value of ‘b’ variable is true and the negation of the value of ‘a’ variable and the value of ‘b’ variable is true by using the logical OR operator.

If the condition is true execute the statement. Multiply the resulted value with 10 integer value. Otherwise, if the condition is false execute the else statement by calling the bintogray() method passing ‘bin’ variable value as argument. Multiply the resulted value with 10 integer value and return the result to the called function ‘gray’ variable. Print the binary code of a number into its equivalent gray code using recursion.

advertisement

Runtime Test Cases

```
$ cc pgm21.c
```

```
$ a.out
Enter a binary number: 1011101
The gray code of 1011101 is 1110011
```

137. C Program to Find Product of 2 Numbers using Recursion

[« Prev](#)

[Next »](#)

This is a C program to find product of 2 numbers using recursion.

Problem Description

This C program finds the product of 2 numbers using recursion.

Problem Solution

This C program using recursion, finds the product of 2 numbers without using the multiplication operator.

advertisement

Program/Source Code

Here is the source code of the C program to display a linked list in reverse. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find Product of 2 Numbers using Recursion
 */
#include <stdio.h>

int product(int, int);

int main()
{
    int a, b, result;

    printf("Enter two numbers to find their product: ");
    scanf("%d%d", &a, &b);
    result = product(a, b);
```

```
printf("Product of %d and %d is %d\n", a, b, result);
      return 0;
}

int product(int a, int b)
{
    if (a < b)
    {
        return product(b, a);
    }
    else if (b != 0)
    {
        return (a + product(a, b - 1));
    }
    else
    {
        return 0;
    }
}
```

Program Explanation

In this C program, reading two numbers using ‘a’ and ‘b’ variables respectively. The product() function is used to find the product of two numbers. Nested if else condition statement is used to check the value of ‘a’ variable is less than the value of ‘b’ variable.

advertisement

If the condition is true then execute the statement. Compute the summation of the value of ‘a’ variable with the value. Otherwise, if the condition is false then execute else if condition statement. Check the condition that the value of ‘b’ variable is not equal to 0.

If the condition is true then execute the statement. Otherwise, if the condition is false then execute the else statement and return the null. Print the product of two numbers.

Runtime Test Cases

```
$ cc pgm20.c
$ a.out
Enter two numbers to find their product: 176 340
Product of 176 and 340 is 59840
```

138. C Program to Shutdown or Turn Off the Computer in Linux

[« Prev](#)

[Next »](#)

This is a C program to shutdown or turn off the computer in linux.

Problem Description

This program is used to shutdown or turn off the computer in linux.

Problem Solution

1. Use function system(“shutdown -P now”) to shutdown or turn off the computer in linux.
2. Exit.

advertisement

Program/Source Code

Here is source code of the C Program to shutdown or turn off the computer in linux. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Shutdown or Turn Off the Computer in Linux.
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     system("shutdown -P now");
9.     return 0;
10.}
```

Program Explanation

1. Use function system(“shutdown -P now”) in the main function to shutdown or turn off the computer in linux.
2. Exit.

Runtime Test Cases

shutdown: Need to be root

139. C Program that Merges Lines Alternatively from 2 Files & Print Result

[« Prev](#)

[Next »](#)

This C Program merges lines alternatively from 2 files & print result.

Here is source code of the C Program to merge lines alternatively from 2 files & print result. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program that Merges Lines Alternatively from 2 Files & Print Result
3. */
4. #include<stdio.h>
5. main()
6. {
7.     char file1[10], file2[10];
8.
9.     puts("enter the name of file 1"); /*getting the names of file to be concatenated*/
10.    scanf("%s", file1);
11.    puts("enter the name of file 2");
12.    scanf("%s", file2);
13.    FILE *fptr1, *fptr2, *fptr3;
14.    fptr1=fopen(file1, "r"); /*opening the files in read only mode*/
15.    fptr2=fopen(file2, "r");
16.    fptr3=fopen("merge2.txt", "w+"); /*opening a new file in write,update mode*/
17.    char str1[200];
18.    char ch1, ch2;
19.    int n = 0, w = 0;
20.    while (((ch1=fgetc(fptr1)) != EOF) && ((ch2 = fgetc(fptr2)) != EOF))
21.    {
22.        if (ch1 != EOF) /*getting lines in alternately from two files*/
23.        {
24.            ungetc(ch1, fptr1);
25.            fgets(str1, 199, fptr1);
26.            fputs(str1, fptr3);
27.            if (str1[0] != 'n')
28.                n++; /*counting no. of lines*/
29.        }
30.        if (ch2 != EOF)
31.        {
32.            ungetc(ch2, fptr2);
33.            fgets(str1, 199, fptr2);
34.            fputs(str1, fptr3);
35.            if (str1[0] != 'n')
```

```

36.         n++;      /*counting no.of lines*/
37.     }
38. }
39. rewind(fp3);
40. while ((ch1 = fgetc(fp3)) != EOF)    /*countig no.of words*/
41. {
42.     ungetc(ch1, fp3);
43.     fscanf(fp3, "%s", str1);
44.     if (str1[0] != ' ' || str1[0] != 'n')
45.         w++;
46. }
47. fprintf(fp3, "\n\n number of lines = %d n number of words is = %d\n", n, w - 1);
48. /*appendig comments in the concatenated file*/
49. fclose(fp1);
50. fclose(fp2);
51. fclose(fp3);
52. }
```

advertisement

```

$ cc pgm51.c
$ a.out
enter the name of file 1
c.txt
enter the name of file 2
a.txt
$ vi merge2.txt
```

All participants will be provided with 1:1 Linux Systems **for** Lab work. If participants want, they can bring their own laptops with Linux **in** it. This **enable** them to **do** lot of quality assignments even Sanfoundry internship programs are great learning opportunities.

Students with proven credentials only are enrolled **for** this program and the duration of these programs ranges from 2-6 months full t after the classes are over. If they have Windows, we **install** virtualization software and Ubuntu Linux virtual appliance on top of windows system.

ime. Student must be passionate about Technology topics. As part of Sanfoundry's **biggest open learning initiative**, we are looking **for interns (student or working professional) in following technolog**

```

number of lines = 4
number of words is = 114
```

140. C Program to Find the Number of Lines in a Text File

[« Prev](#)

[Next »](#)

This C Program displays the number of lines in a text file.

Here is source code of the C Program to find the number of lines in a text file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Find the Number of Lines in a Text File
3.  */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     FILE *fileptr;
9.     int count_lines = 0;
10.    char filechar[40], chr;
11.
12.    printf("Enter file name: ");
13.    scanf("%s", filechar);
14.    fileptr = fopen(filechar, "r");
15.    //extract character from file and store in chr
16.    chr = getc(fileptr);
17.    while (chr != EOF)
18.    {
19.        //Count whenever new line is encountered
20.        if (chr == '\n')
21.        {
22.            count_lines = count_lines + 1;
23.        }
24.        //take next character from file.
25.        chr = getc(fileptr);
26.    }
27.    fclose(fileptr); //close file.
28.    printf("There are %d lines in %s in a file\n", count_lines, filechar);
29.    return 0;
30. }
```

advertisement

```
$ cc pgm49.c
$ a.out
Enter file name: pgm2.c
There are 43 lines in pgm2.c in a file
```

141. C Program to Display its own Source Code as its Output

[« Prev](#)

[Next »](#)

This is a C Program to display its own source code as its output.

Problem Description

This program displays its own source code as its output.

Problem Solution

1. Display the content from the same file you are writing the source code.

advertisement

Program/Source Code

Here is source code of the C Program to display its own source code as its output. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Display its own Source Code as its Output
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     FILE *fp;
9.     char ch;
10.
11.    fp = fopen(FILE,"r");
12.    do
13.    {
14.        ch = getc(fp);
15.        putchar(ch);
16.    }
17.    while (ch != EOF);
18.    fclose(fp);
19.    return 0;
20.}
```

Program Explanation

1. Open the file you are currently writing using statement `fopen(FILE,"r")` and assign it to the pointer fp.
2. Scan the every character of the file and store it in the variable ch. Print it using statement `putchar(ch)`.

3. Do step 2 until EOF (end of file).
4. Then close the file and exit.

advertisement

Runtime Test Cases

Output:

```
/*
 * C Program to display its own source code as its output
 */
#include <stdio.h>

int main()
{
    FILE *fp;
    char ch;

    fp = fopen(__FILE__,"r");
    do
    {
        ch = getc(fp);
        putchar(ch);
    }
    while (ch != EOF);
    fclose(fp);
    return 0;
}
```

142. C Program to Find the Odd Element given an Array with only two Different Element

[« Prev](#)

[Next »](#)

This is a C Program to find the odd element given an array with only two different elements.

Problem Description

This C Program finds odd element in a given array with only two different element.

Problem Solution

Print the odd element with only two different elements in an array.

advertisement

Program/Source Code

Here is source code of the C Program to find the odd element given an array with only two different element. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Odd Element given an Array with only two Different Element
 */
#include <stdio.h>

void printodd(int array[], int size)
{
    int xor2 = array[0]; /* Will hold XOR of two odd occurring elements */
    int set;
    int i;
    int n = size - 2;
    int x = 0, y = 0;

    /* The xor will basically be xor of two odd occurring elements */
    for (i = 1; i < size; i++)
        xor2 = xor2 ^ array[i];
    /* Get one set rightmost bit in the xor2. */
    set = xor2 & ~(xor2 - 1);
    /* Now divide elements in two sets: */
    for (i = 0; i < size; i++)
    {
        /* XOR of first set is finally going to hold one odd occurring number x */
        if (array[i] & set)
            x = x ^ array[i];
    }
}
```

```

/* XOR of second set is finally going to hold the other odd occurring number y */
else
    y = y ^ array[i];
}
printf("\n The ODD elements are %d & %d ", x, y);
}

int main()
{
    int array[] = {10, 3, 2, 10, 2, 8, 8, 7};
    int arr_size = sizeof(array) / sizeof(array[0]);
    printodd(array, arr_size);
    getchar();
    return 0;
}

```

Program Explanation

In this C Program, we have defined the array[] variable elements. Using ‘array_size’ variable compute the division of size of previous array value by the next value of array[] variable element.

advertisement

The printodd() function is used to find the odd element given an array with only two different elements. The xor2 variable is used to hold XOR of two odd occurring elements. For loop is used to compute, the xor will basically be xor of two odd occurring elements.

Divide the elements in two sets, XOR of first set is finally going to hold one odd occurring number x. The XOR of second set is finally going to hold the other odd occurring number y.

Runtime Test Cases

```
$ cc pgm87.c
$ a.out
```

The ODD elements are 7 & 3

143. C Program to Find the Sum of Series $1^3 + 2^3 + 3^3 + \dots + n^3$

[« Prev](#)

[Next »](#)

This C Program calculates the Sum of Series $1^3 + 2^3 + 3^3 + \dots + n^3$. Sum of the series $1^3 + 2^3 + 3^3 + \dots + n^3 = (n(n + 1) / 2)^2$

Here is source code of the C Program to Find the Sum of Series $1^3 + 2^3 + 3^3 + \dots + n^3$. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Sum of Series  $1^3 + 2^3 + 3^3 + \dots + n^3$ 
3. */
4. #include <stdio.h>
5. #include <math.h>
6.
7. int main()
8. {
9.     int number, i;
10.    int sum = 0;
11.
12.    printf("Enter the maximum values of series n: ");
13.    scanf("%d", &number);
14.    sum = pow(((number * (number + 1)) / 2),2);
15.    printf("Sum of the series : ");
16.    for (i = 1; i <= number; i++)
17.    {
18.        if (i != number)
19.            printf("%d^3 + ", i);
20.        else
21.            printf("%d^3 = %d ", i, sum);
22.    }
23.    return 0;
24.}
```

advertisement

Output:

```
$ cc pgm17.c
$ a.out
```

```
Enter the maximum values of series n: 5
Sum of the series :  $1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225$ 
```

144. C Program to Display Floyd's triangle

[« Prev](#)

[Next »](#)

This is a C Program to display floyd's triangle.

Problem Description

This C Program displays floyd's triangle.

Problem Solution

It displays floyd's triangle as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to display floyd's triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Display Floyd's triangle
 */
#include <stdio.h>

main()
{
    int i, j, k = 1;

    printf("floyds triangle is\n");
    for( i = 1; k <= 20; ++i )
    {
        for( j = 1; j <= i; ++j )
            printf( "%d ", k++ );
        printf( "\n\n" );
    }
    return 0;
}
```

Program Explanation

This C program is used to print Floyd's triangle. Floyd's triangle is a right-angled triangular array of natural numbers. It is defined by filling the rows of the triangle with consecutive numbers, starting with a 1 in the top left corner: 1. Number of rows of Floyd's triangle to print is entered by the user. For loop is used to print the output of the program.

advertisement

Runtime Test Cases

Output:
\$ cc pgm36.c
\$ a.out
floyds triangle is
1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21

145. C Program to Find Area of Trapezium

[« Prev](#)

[Next »](#)

This is a C Program to find area of trapezium.

Problem Description

This C Program calculates the area of trapezium.

Problem Solution

The formula used in this program are $\text{Area} = (1/2) * (a + b) * h$ where a and b are the 2 bases of trapezium & h is the height.

advertisement

Program/Source Code

Here is source code of the C Program to Find the area of a right angled triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Area of Trapezium
 */
#include <stdio.h>

int main()
{
    float a, b, h;
    float area;

    printf("Enter the value for two bases & height of the trapezium: \n");
    scanf("%f%f%f", &a, &b, &h);
    area = 0.5 * (a + b) * h;
    printf("Area of the trapezium is: %.3f", area);
    return 0;
}
```

Program Explanation

In this C program, library function defined in `<math.h>` header file is used to compute mathematical functions. We are reading the two bases and height of a trapezium using ‘ a ’, ‘ b ’ and ‘ h ’ variable. To find the surface area, the following formulas is used.

advertisement

$\text{Area} = (1/2) * (a + b) * h.$

Runtime Test Cases

Output:

```
$ cc pgm25.c
```

```
$ a.out
```

```
Enter the value for two bases and height of the trapezium :
```

```
10 15 20
```

```
Area of the trapezium is: 250.000
```

146. C Program to Convert Octal to Binary

[« Prev](#)

[Next »](#)

This is a C program to Convert Octal to Binary.

Problem Description

This program takes a octal number as input and converts it into binary.

Problem Solution

1. Take a octal number as input.
2. Print the binary value of each digit of a octal number. Use switch statement and while loop to do this.

advertisement

Program/Source Code

Here is source code of the C program to Convert Octal to Binary. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Convert Octal to Binary
3.  */
4. #include <stdio.h>
5. #define MAX 1000
6.
7. int main()
8. {
9.     char octalnum[MAX];
10.    long i = 0;
11.
12.    printf("Enter any octal number: ");
13.    scanf("%s", octalnum);
14.    printf("Equivalent binary value: ");
15.    while (octalnum[i])
16.    {
17.        switch (octalnum[i])
18.        {
19.            case '0':
20.                printf("000"); break;
21.            case '1':
22.                printf("001"); break;
23.            case '2':
24.                printf("010"); break;
25.            case '3':
26.                printf("011"); break;
27.            case '4':
28.                printf("100"); break;
29.            case '5':
30.                printf("101"); break;
31.            case '6':
```

```
32.     printf("110"); break;
33. case '7':
34.     printf("111"); break;
35. default:
36.     printf("\n Invalid octal digit %c ", octalnum[i]);
37.     return 0;
38. }
39. i++;
40. }
41. return 0;
42. }
```

Program Explanation

1. Take a octal number as input and store it in the array octalnum.
2. Using switch statement access each digit of a octal number and print its equivalent binary value in a 3 bit fashion. For example: for 0, print its binary value as 000.
3. Do step 2 under a while loop.
4. Exit.

advertisement

Runtime Test Cases

Output:

Enter any octal number: a

Equivalent binary value:

Invalid octal digit a

Enter any octal number: 160

Equivalent binary value: 001110000

147. C Program to Find Union & Intersection of 2 Arrays

[« Prev](#)

[Next »](#)

This C Program finds union & intersection of 2 arrays. Union here refers to the set of all the elements of the 2 arrays. Intersection here refers to the set of elements which are in both the arrays.

Here is source code of the C Program to find union & intersection of 2 arrays. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Find Union & Intersection of 2 Arrays
3.  */
4. #include <stdio.h>
5. #define SIZE 5
6.
7. void get_value(int arr[]);
8. void print_value(int arr[], int n);
9. void function_sort(int arr[]);
10.int find_intersection(int array1[], int array2[], int intersection_array[]);
11.int find_union(int array1[], int array2[], int union_array[]);
12.
13.void main()
14.{
15.    int array1[SIZE], array2[SIZE], intersection_array[SIZE], union_array[SIZE*2];
16.    int num_elements;
17.
18.    //input elements of Array1
19.    printf("\n Enter the elements of Array 1: n");
20.    get_value(array1);
21.    printf("\n\n Elements of Array 1: ");
22.    print_value(array1, SIZE);
23.
24.    //Sort array 1
25.    function_sort(array1);
26.    printf("nnSorted elements of Array 1: ");
27.    print_value(array1, SIZE);
28.
29.    //input elements of Array2
30.    printf("nnEnter the elements of Array 2: n");
31.    get_value(array2);
32.    printf("\n\n Elements of Array 2: ");
33.    print_value(array2, SIZE);
34.
35.    //Sort array 2
36.    function_sort(array2);
37.    printf("\n\nSorted elements of Array 2: ");
38.    print_value(array2, SIZE);
39.
40.    //Find Intersection
41.    num_elements = find_intersection(array1, array2, intersection_array);
42.    printf("\n\n Intersection is: ");
43.    print_value(intersection_array, num_elements);
44.
45.    //Find Union
```

```

46. num_elements = find_union(array1, array2, union_array);
47. printf("\n\n Union is: ");
48. print_value(union_array, num_elements);
49. }
50.
51.void get_value(int arr[])
52.{
53. int i, j;
54. for (i = 0; i < SIZE; i++)
55. {
56.     j = i + 1;
57.     printf("\n Enter element %d: ", j);
58.     scanf("%d", &arr[i]);
59. }
60.}
61.
62.void print_value(int arr[], int n)
63.{
64. int i;
65. printf("{ ");
66. for (i = 0; i < n; i++)
67. {
68.     printf("%d ", arr[i]);
69. }
70. printf("}");
71.}
72.
73.void function_sort(int arr[])
74.{
75. int i, j, temp, swapping;
76.
77. for (i = 1; i < size; i++)
78. {
79.     swapping = 0;
80.     for (j = 0; j < size-i; j++)
81.     {
82.         if (arr[j] > arr[j+1])
83.         {
84.             temp = arr[j];
85.             arr[j] = arr[j + 1];
86.             arr[j + 1] = temp;
87.             swapping = 1;
88.         }
89.     }
90.     if(swapping == 0)
91.     {
92.         break;
93.     }
94. }
95.}
96.
97.int find_intersection(int array1[], int array2[], int intersection_array[])
98.{
99. int i = 0, j = 0, k = 0;
100. while ((i < size) && (j < size))
101. {
102.     if (array1[i] < array2[j])
103.     {
104.         i++;
105.     }
106.     else if (array1[i] > array2[j])
107.     {
108.         j++;
109.     }
110.     else
111.     {

```

```

112.     intersection_array[k] = array1[i];
113.     i++;
114.     j++;
115.     k++;
116.   }
117. }
118. return(k);
119.}
120.
121.int find_union(int array1[], int array2[], int union_array[])
122.{
123. int i = 0, j = 0, k = 0;
124. while ((i < SIZE) && (j < SIZE))
125. {
126.   if (array1[i] < array2[j])
127.   {
128.     union_array[k] = array1[i];
129.     i++;
130.     k++;
131.   }
132.   else if (array1[i] > array2[j])
133.   {
134.     union_array[k] = array2[j];
135.     j++;
136.     k++;
137.   }
138.   else
139.   {
140.     union_array[k] = array1[i];
141.     i++;
142.     j++;
143.     k++;
144.   }
145. }
146. if (i == SIZE)
147. {
148.   while (j < SIZE)
149.   {
150.     union_array[k] = array2[j];
151.     j++;
152.     k++;
153.   }
154. }
155. else
156. {
157.   while (i < SIZE)
158.   {
159.     union_array[k] = array1[i];
160.     i++;
161.     k++;
162.   }
163. }
164. return(k);
165.}

```

advertisement

```
$ cc pgm98.c
$ a.out
```

Enter the elements of Array 1:

Enter element 1: 12

Enter element 2: 34

Enter element 3: 23

Enter element 4: 56

Enter element 5: 45

Elements of Array 1: { 12 34 23 56 45 }

Sorted elements of Array 1: { 12 23 34 45 56 }

Enter the elements of Array 2:

Enter element 1: 34

Enter element 2: 56

Enter element 3: 12

Enter element 4: 78

Enter element 5: 66

Elements of Array 2: { 34 56 12 78 66 }

Sorted elements of Array 2: { 12 34 56 66 78 }

Intersection is: { 12 34 56 }

Union is: { 12 23 34 45 56 66 78 }

148. C Program to Print a Semicolon without using a Semicolon anywhere in the Code

« [Prev](#)

[Next](#) »

This is a C Program to print a semicolon without using a semicolon anywhere in the code.

Problem Description

This program prints a semicolon without using a semicolon anywhere in the code.

Problem Solution

1. Print the ASCII value of semicolon and exit.

advertisement

Program/Source Code

Here is source code of the C Program to print a semicolon without using a semicolon anywhere in the code. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Print a Semicolon without using a Semicolon
3. * anywhere in the code
4. */
5. #include <stdio.h>
6.
7. int main(void)
8. {
9. //59 is the ascii value of semicolon
10. if (printf("%c ", 59))
11. {
12. }
13. return 0;
14. }
```

Program Explanation

1. The ASCII value of semicolon is 59.
2. Print this ASCII value and exit.

advertisement

Runtime Test Cases

;

149. C Program to Find the Perimeter of a Circle, Rectangle and Triangle

[« Prev](#)

[Next »](#)

This is a C Program to find the perimeter of a circle, rectangle and triangle.

Problem Description

This C Program calculates the perimeter of a circle, rectangle and triangle.

Problem Solution

This program is used to find the perimeter of a circle, rectangle and triangle. The formula used in this program are

perimeter of rectangle: $2 * (a + b)$

perimeter of General triangle: $a + b + c$

perimeter of Equilateral triangle: $3 * a$

perimeter of Right angled triangle: $\text{width} + \text{height} + \sqrt{\text{width}^2 + \text{height}^2}$

perimeter of circle: $2 * \pi * r$

advertisement

Program/Source Code

Here is source code of the C Program to Find the perimeter of a circle, rectangle & triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Perimeter of a Circle, Rectangle and Triangle
 */
#include <stdio.h>
#include <math.h>

int main()
{
    float radius, length, width, a, b, c, height;
    int n;
    float perimeter;

    //Perimeter of rectangle
    printf("\n Perimeter of rectangle \n");
    printf("-----\n");
    printf("\n Enter width and length of the rectangle : ");
    scanf("%f%f", &width,& length);
    perimeter = 2 * (width + length);
    printf("Perimeter of rectangle is: %.3f", perimeter);

    //Perimeter of triangle
```

```

printf("\n Perimeter of triangle\n");
printf("-----\n");
printf("\n Enter the size of all sides of the triangle : ");
scanf("%f%f%f", &a, &b, &c);
perimeter = a + b + c;
printf("Perimeter of triangle is: %.3f", perimeter);

//Perimeter of circle
printf("\n Perimeter of circle \n");
printf("-----\n");
printf("\n Enter the radius of the circle : ");
scanf("%f", &radius);
perimeter = 2 * (22 / 7) * radius;
printf("Perimeter of circle is: %.3f", perimeter);

//Perimeter of equilateral triangle
printf("\n Perimeter of equilateral triangle \n");
printf("-----\n");
printf("\n Enter any side of the equilateral triangle : ");
scanf("%f", &a);
perimeter = 3 * a;
printf("Perimeter of equilateral triangle is: %.3f", perimeter);

//Perimeter of right angled triangle
printf("\n Perimeter of right angled triangle \n");
printf("-----\n");
printf("\n Enter the width and height of the right angled triangle : ");
scanf("%f%f", &width, &height);
perimeter = width + height + sqrt(width * width + height * height);
printf("Perimeter of right angled triangle is: %.3f", perimeter);
return 0;
}

```

Program Explanation

This C program is used to find the perimeter of a circle, rectangle and triangle. We are reading the value for ‘width’ and ‘length’ variables respectively. Compute the perimeter of a rectangle. The following formula is used

advertisement

$\text{Perimeter} = 2 * (\text{width} + \text{length})$.

We are reading the values for ‘a’, ‘b’, ‘c’ variables respectively. Compute the perimeter of the triangle, the following the formula is used.

$\text{Perimeter} = a + b + c$.

advertisement

We are reading the value for ‘radius’ variable. Compute the perimeter of circle, the following formula is used

$\text{Perimeter} = 2 * (22/7) * \text{radius}$.

We are reading the value for ‘a’ variable. Compute the perimeter of a equilateral triangle, the following formula is used.

$\text{Perimeter} = 3 * a$.

advertisement

We are reading the values for ‘width’ and ‘height’ variables respectively. Compute the perimeter of Right angled triangle, the following formula is used

Perimeter = width + height + sqrt((width * width) + (height * height)).

Runtimme Test Cases

Output:

\$ cc pgm32.c -lm

\$ a.out

Perimeter of rectangle

Enter width and length of the rectangle : 12 13

Perimeter of rectangle is: 50.000

Perimeter of triangle

Enter the size of all sides of the triangle : 12 16 18

Perimeter of triangle is: 46.000

Perimeter of circle

Enter the radius of the circle : 10

Perimeter of circle is: 60.000

Perimeter of equilateral triangle

Enter any side of the equilateral triangle : 19 34

Perimeter of equilateral triangle is: 57.000

Perimeter of right angled triangle

Enter the width and height of the right angled triangle : 5 7

Perimeter of right angled triangle is: 73.366

150. C Program to Check Array bounds while Inputting Elements into the Array

« [Prev](#)

[Next](#) »

This is a C program check array bounds while inputing elements into the array.

Problem Description

This is a C program which implements an array and checks array bounds while inputing elements into the array.

Problem Solution

1. Create an array of some fixed capacity.
2. Run a for loop more than the array capacity times, inputing array elements.
3. Print array elements using for loop, where the elements entered by user are printed as it is, but rest printed numbers will be garbage values.

advertisement

Program/Source Code

Here is source code of the C Program to check array bounds while inputing elements into the array. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1.  /*
2.   * C Program to Check Array bounds while Inputing Elements into the Array
3.   */
4.
5. #include <stdio.h>
6. int main(void)
7. {
8.
9.     int array[5], b, c;
10.    for (b = 0; b < 10 && (scanf("%d", &c)); b++)
11.        array[b] = c;
12.
13.    for (b = 0; b < 15; b++)
14.        printf("%d ", array[b]);
15.
16.    return 0;
17. }
```

Program Explanation

1. Declare an array of some fixed capacity, lets say 5.
2. Using a for loop, take input from users each array element. Loop in this program runs more than the size of the array.
3. As the expression array[position] equals *(array + position), therefore the loop continues to go beyond array size and defining values at the required position.

4. At the time of printing, we have executed the for loop 15 times, whereas the size of the array is 5 and we have defined the elements from start position of the array to the 10th position of the array.

5. Therefore, we get first 10 numbers as whatever we entered, but after that since we did not enter any number, so already stayed garbage values are written.

advertisement

Runtime Test Cases

```
12  
23  
56  
12  
14  
23  
12 23 56 12 14 23 6 134513824 0 -1081194040 11672807 1 -1081193996 -1081193988 -1216161720
```

151. C Program to Find the Volume and Surface Area of Cuboids

[« Prev](#)

[Next »](#)

This is a C Program to find the volume and surface area of cuboids.

Problem Description

This C Program calculates the volume and surface area of cuboids.

Problem Solution

The formula used in this program are $\text{surfarea} = 2(w * l + l * h + h * w)$ where w is width, l is length and h is a height of the cuboids. $\text{volume} = \text{width} * \text{length} * \text{height}$.

advertisement

Program/Source Code

Here is source code of the C Program to Find the volume and surface area of cuboids. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Volume and Surface Area of Cuboids
 */
#include <stdio.h>
#include <math.h>

int main()
{
    float width, length, height;
    float surfacearea, volume, space_diagonal;

    printf("Enter value of width, length & height of the cuboids:\n");
    scanf("%f%f%f", &width, &length, &height);
    surfacearea = 2 * (width * length + length * height +
    height * width);
    volume = width * length * height;
    space_diagonal = sqrt(width * width + length * length +
    height * height);
    printf("Surface area of cuboids is: %.3f", surfacearea);
    printf("\n Volume of cuboids is : %.3f", volume);
    printf("\n Space diagonal of cuboids is : %.3f", space_diagonal);
    return 0;
}
```

Program Explanation

In this C program, library function defined in `<math.h>` header file is used to compute mathematical functions. We are reading the ‘width’, ‘length’ and ‘height’ values of cuboids. To find the surface area and the volume, the following formulas are used.

advertisement

Surface area= $2(\text{width} * \text{length} + \text{length} * \text{height} + \text{height} * \text{width})$

Volume = $\text{width} * \text{length} * \text{height}$

Diagonal = $\sqrt{\text{width} * \text{width} + \text{length} * \text{length} + \text{height} * \text{height}}$.

advertisement

Runtime Test Cases

Output:

\$ cc pgm28.c -lm

\$ a.out

Enter value of width, length & height of the cuboids :

22 23 24

Surface area of cuboids is: 3172.000

Volume of cuboids is : 12144.000

Space diagonal of cuboids is : 39.862

152. C Program to Find Area of a Right Angled Triangle

[« Prev](#)

[Next »](#)

This is a C Program to find area of a right angled triangle.

Problem Description

This C Program calculates the area of a right angled triangle.

Problem Solution

The formula used in this program are $\text{Area} = (1/2) * \text{height} * \text{width}$.

advertisement

Program/Source Code

Here is source code of the C Program to Find the area of a right angled triangle. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Area of a Right Angled Triangle
 */
#include <stdio.h>

int main()
{
    float height, width;
    float area;

    printf("Enter height and width of the given triangle:\n");
    scanf("%f%f", &height, &width);
    area = 0.5 * height * width;
    printf("Area of right angled triangle is: %.3f\n", area);
    return 0;
}
```

Program Explanation

In this C program, library function defined in `<math.h>` header file is used to compute mathematical functions. We are reading the ‘height’ and ‘width’ of a triangle. To find the surface area, the following formulas is used.

advertisement

$\text{Area} = (1/2) * \text{height} * \text{width}$.

Runtime Test Cases

```
Output:  
$ cc pgm24.c  
$ a.out  
Enter height and width of the given triangle:  
10 15  
Area of right angled triangle is: 75.000
```

153. C Program to Find the Sum of G.P Series

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of G.P series.

Problem Description

This C Program calculates the sum of G.P series.

Problem Solution

This program is used to find the sum of the geometric progression series. Here G.P stands for geometric progression. A geometric progression, or GP, is a sequence where each new term after the first is obtained by multiplying the preceding term by a constant r, called the common ratio. The formula used in this program are $T_n = a * (r^{n-1})$. where T_n is the last term of a finite sequence. $S_n = a(1 - r^n) / (1 - r)$ where S_n is the sum of n terms.

advertisement

Program/Source Code

Here is source code of the C Program to Find the the sum of G.P series. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*  
 * C Program to Find the Sum of G.P Series  
 */  
  
#include <stdio.h>  
#include <math.h>  
  
int main()  
{  
    float a, r, i, last_term, sum = 0;  
    int n;  
  
    printf("Enter the first term of the G.P. series: ");  
    scanf("%f", &a);  
    printf("Enter the total numbers in the G.P. series: ");  
    scanf("%d", &n);  
    printf("Enter the common ratio of G.P. series: ");  
    scanf("%f", &r);  
    sum = (a * (1 - pow(r, n + 1))) / (1 - r);  
    last_term = a * pow(r, n - 1);  
    printf("last_term term of G.P.: %f", last_term);  
    printf("\n Sum of the G.P.: %f", sum);  
    return 0;  
}
```

Program Explanation

In this program, we are reading the first term of the G.P. series using ‘a’ variable and the total numbers in the G.P. Series using ‘n’ variable and the common ratio of G.P series using ‘r’ variable.

advertisement

A geometric progression, or GP, is a sequence where each new term after the first is obtained by multiplying the preceding term by a constant r, called the common ratio. The formula used in this program is

$$T_n = (a * (1 - pow(r, n + 1))) / (1 - r).$$

Where, T_n is the last term of a finite sequence.

advertisement

$$S_n = a * (1 - pow(r, n - 1))$$

Where, S_n is the sum of n terms.

Runtime Test Cases

Output:

```
$ cc pgm22.c -lm
$ a.out
Enter the first term of the G.P. series: 3
Enter the total numbers in the G.P. series: 7
Enter the common ratio of G.P. series: 2
last_term term of G.P.: 192.000000
Sum of the G.P.: 765.000000
```

154. C Program to Input a String & Store their Ascii Values in an Integer Array & Print the Array

[« Prev](#)

[Next »](#)

This is a C Program to input a string & store their Ascii Values in an Array & print the Array

Problem Description

This program will take an input of string & store their ASCII values and print them.

Problem Solution

1. Create an array of characters and take its size from users as an input.
2. Enter a string.
3. Using a loop, scan each element(character) of the string and print its equivalent ASCII value, increment the value of iterator used to access the position of characters of the string.

advertisement

Program/Source Code

Here is source code of the C Program to input a string & store their ascii values and print them. The program is successfully compiled and tested using Turbo C compiler in windows environment. The program output is also shown below.

```
1. /*
2.  * C Program to Input a String & Store their Ascii Values in an Integer Array & Print the Array
3.  */
4.
5. #include <stdio.h>
6. void main()
7. {
8.
9.     char string[20];
10.    int n, count = 0;
11.
12.    printf("Enter the no of characters present in an array \n ");
13.    scanf("%d", &n);
14.
15.    printf(" Enter the string of %d characters \n ", n);
16.    scanf("%s", string);
17.
18.    while (count < n)
19.    {
20.        printf(" %c = %d\n ", string[count], string[count] );
21.        ++ count ;
22.    }
23.
24. }
```

Program Explanation

1. Declare an array of characters of some fixed size, 20.
2. Take size from users as an input.
3. Now, enter a string again as an input, store it in character array declared above.
4. Run a for loop, which will scan each array element (i.e character), printing its equivalent ASCII code using %d notation and the character itself using %c inside printf() function.
5. Increment iterator, count after printf() statement to access next position.
6. Exit

advertisement

Runtime Test Cases

Enter the no of characters present in an array

10

Enter the string of 10 characters

sanfoundry

```
s = 115  
a = 97  
n = 110  
f = 102  
o = 111  
u = 117  
n = 110  
d = 100  
r = 114  
y = 121
```

155. C program to Calculate the Value of nPr

[« Prev](#)

[Next »](#)

This is a C Program to calculate the value of nPr.

Problem Description

This C Program calculates the value of nPr.

Problem Solution

Here we need to find all possible rearrangement of the element i.e all the possible permutation value. A permutation is a re-arrangement of elements of a set. Any duplications of the collected elements in different orders are allowed. A permutation therefore tends to be a large number.

advertisement

Program/Source Code

Here is source code of the C program to calculate the Value of nPr. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to Calculate the Value of nPr
 */
#include <stdio.h>

void main(void)
{
    printf("%d\n", fact(8));
    int n, r;
    printf("Enter value for n and r\n");
    scanf("%d%d", &n, &r);
    int npr = fact(n) / fact(n - r);
    printf("\n Permutation values is = %d", npr);
}

int fact(int x)
{
    if (x <= 1)
        return 1;
    return x * fact(x - 1);
}
```

Program Explanation

In this C program, we are reading the two integer values using ‘n’ and ‘r’ variables respectively. The fact() function is used to find all possible rearrangement of the elements. A permutation is a re-arrangement of elements of a set. Any duplication of the collected elements in different orders is allowed. A permutation therefore tends to be a large number.

advertisement

If condition statement is used to check the integer value is less than or equal to 1. If the condition is true, then execute the statement and return the value as 1. Otherwise, if the condition is false then execute the else statement.

Compute the integer value with the next previous value i.e if the integer value is 3. Multiply the value as $3*2$ then the resultant value 6 with 1 and return the value to ‘npr’ variable. Divide the value of ‘integer’ variable by fact(). Compute the difference of the value of ‘integer’ variable by the value of ‘r’ power variable. Print the value of nPr using the printf statement.

Runtime Test Cases

Output:

```
$ cc pgm13.c
```

```
$ a.out
```

```
40320
```

```
Enter value for n and r
```

```
5 4
```

```
Permutation values is = 120
```

156. C Program to Convert Roman Number to Decimal Number

[« Prev](#)

[Next »](#)

This is a C Program to Convert Roman Number to Decimal Number.

Problem Description

This program takes a roman number as input and converts it to decimal number.

Problem Solution

1. Take a roman number as input.
2. Using switch statement define the value of each roman digit.
3. Through switch statement access each digit of a roman number and compute the value.
4. Print the value and exit.

advertisement

Program/Source Code

Here is source code of the C program to Convert Roman Number to Decimal Number. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. /*
3. * C Program to Convert Roman Number to Decimal Number
4. */
5.
6. #include<stdio.h>
7. #include<string.h>
8.
9. int digit(char);
10.
11.int main(){
12.
13. char roman_Number[1000];
14. int i=0;
15. long int number =0;
16.
17. printf("Enter any roman number (Valid digits are I, V, X, L, C, D, M): \n");
18. scanf("%s",roman_Number);
19.
20. while(roman_Number[i]){
21.
22. if(digit(roman_Number[i]) < 0){
23. printf("Invalid roman digit : %c",roman_Number[i]);
24. return 0;
25. }
26.
27. if((strlen(roman_Number) -i) > 2){
28. if(digit(roman_Number[i]) < digit(roman_Number[i+2])){
29. printf("Invalid roman number");
30. return 0;
31. }
```

```

32. }
33.
34. if(digit(roman_Number[i]) >= digit(roman_Number[i+1]))
35.     number = number + digit(roman_Number[i]);
36. else{
37.     number = number + (digit(roman_Number[i+1]) - digit(roman_Number[i]));
38.     i++;
39. }
40. i++;
41. }
42.
43. printf("Its decimal value is : %ld",number);
44.
45. return 0;
46.
47. }
48.
49.int digit(char c){
50.
51. int value=0;
52.
53. switch(c){
54.     case 'I': value = 1; break;
55.     case 'V': value = 5; break;
56.     case 'X': value = 10; break;
57.     case 'L': value = 50; break;
58.     case 'C': value = 100; break;
59.     case 'D': value = 500; break;
60.     case 'M': value = 1000; break;
61.     case '\0': value = 0; break;
62.     default: value = -1;
63. }
64.
65. return value;
66.

```

Program Explanation

1. Take a roman number as input and store it in the array roman_Number.
2. In the function digit(), define the value of each digit of the roman number inside the switch statement and return the same.
3. Using while statement access each digit of the input number.
4. Firstly check if the current roman digit's value is less than zero. If it is, then print the output as “Invalid roman digit”.
5. If not, then check if the value of current roman digit is greater or equal to its next digit's value. If it is, then increment the variable number with the value of current roman digit.
6. Otherwise, subtract the value of current roman digit from the value of its next roman digit and increment the variable number with the obtained value.
7. Print the variable number as output.

advertisement

Runtime Test Cases

Output:

Enter any roman number (Valid digits are I, V, X, L, C, D, M):

XVII

Its decimal value is: 17

157. C program to Calculate the value of nCr

[« Prev](#)

[Next »](#)

This is a C Program to calculate the value of nCr.

Problem Description

This C Program Calculates the value of nCr.

Problem Solution

The algorithm used in this program is $nCr = n! / ((n-r)!r!)$. Here we need to find all the possible combination of the value n and r. A combination is one or more elements selected from a set without regard to the order. The “without regard” means that the collection matters rather than order in combinations, so in the above example, the fact we ABC, ACB, BAC, BCA, CAB, CBA... for combinations, these are all 1 combination of letters A, B and C.

advertisement

Program/Source Code

Here is source code of the C program to Calculate the value of nCr. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C program to Calculate the value of nCr
 */
#include <stdio.h>

int fact(int z);

void main()
{
    int n, r, ncr;

    printf("\n Enter the value for N and R \n");
    scanf("%d%d", &n, &r);
    ncr = fact(n) / (fact(r) * fact(n - r));
    printf("\n The value of ncr is: %d", ncr);
}

int fact(int z)
{
    int f = 1, i;
    if (z == 0)
    {
        return(f);
    }
    else
    {
        for (i = 1; i <= z; i++)
        {
            f = f * i;
        }
        return(f);
    }
}
```

Program Explanation

In this C Program, we are reading the value for ‘n’ and ‘r’ variable to calculate the value of nCr. The algorithm used in this program is $nCr = n! /((n-r)!r!)$. A combination is one or more elements selected from a set without regard to the order.

advertisement

Then ‘ncr’ variable is used to compute $fact(n)/(fact(r)* fact(n - r))$. The fact() function is used to compute the factorial of the value. If-else condition statement is used to check the argument value of ‘z’ variable is equal to 0. If the condition is true then execute the statement.

For loop is used to compute the factorial value. Initialize the value of ‘i’ variable to 1 and check the value of ‘i’ variable is less than or equal to the argument value in ‘z’ variable. If the condition is true then execute the loop. Multiply the value of ‘f’ variable with each integer variable value in ‘i’ variable. Compute the value for nCr and print the value of nCr using printf statement.

Runtime Test Cases

```
Output:  
$ cc pgm12.c  
$ a.out  
Enter the value for N and R  
5 2
```

The value of ncr is: 10

158. C Program to Find & Display Multiplication table

[« Prev](#)

[Next »](#)

This is a C Program to Find & Display Multiplication table.

Problem Description

This program finds & displays the multiplication table of a particular number.

Problem Solution

1. Take a number as input.
2. Using while loop print the number and its product with numbers 1-10 as output.

advertisement

Program/Source Code

Here is source code of the C Program to Find & Display Multiplication table. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find & Display Multiplication table
3. */
4. #include <stdio.h>
5.
6. int main()
7. {
8.     int number, i = 1;
9.
10.    printf(" Enter the Number:");
11.    scanf("%d", &number);
12.    printf("Multiplication table of %d:\n ", number);
13.    printf("-----\n");
14.    while (i <= 10)
15.    {
16.        printf(" %d x %d = %d \n ", number, i, number * i);
17.        i++;
18.    }
19.    return 0;
20.}
```

Program Explanation

1. Take a number as input and store it in the variable number.
2. Using while loop starting from 1 to 10, print the variable number, i and (number * i) as output and exit.

advertisement

Runtime Test Cases

Output:

```
Enter the Number:6
Multiplication table of 6:
-----

```

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

159. C Program for Depth First Binary Tree Search using Recursion

[« Prev](#)

[Next »](#)

The following C program, using recursion, performs a Depth First Search traversal. Depth-first search (DFS) is an algorithm for traversing or searching a tree, tree structure or graph. The concept of backtracking is used in DFS. In this program we are performing DFS on a binary tree. In DFS, the deepest and unvisited node is visited and backtracks to its parent node if no siblings of that node exists.

Conditions: The DFS works on acyclic graph. DFS may fail if it enters a cycle. Care must be taken by not extending a path to a node if it already has.

Here is the source code of the C program to apply DFS on a binary tree recursively. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program for Depth First Binary Tree Search using Recursion
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. struct node
8. {
9.     int a;
10.    struct node *left;
11.    struct node *right;
12.};
13.
14.void generate(struct node **, int);
15.void DFS(struct node *);
16.void delete(struct node **);
17.
18.int main()
19.{
20.    struct node *head = NULL;
21.    int choice = 0, num, flag = 0, key;
22.
23.    do
24.    {
25.        printf("\nEnter your choice:\n1. Insert\n2. Perform DFS Traversal\n3. Exit\nChoice: ");
26.        scanf("%d", &choice);
27.        switch(choice)
28.        {
29.            case 1:
30.                printf("Enter element to insert: ");
31.                scanf("%d", &num);
32.                generate(&head, num);
33.                break;
34.            case 2:
35.                DFS(head);
36.                break;
37.            case 3:
38.                delete(&head);
39.                printf("Memory Cleared\nPROGRAM TERMINATED\n");
40.                break;
41.            default:
42.                printf("Not a valid input, try again\n");
43.        }
44.    } while (choice != 3);
45.    return 0;
46.}
```

```

47.
48.void generate(struct node **head, int num)
49.{
50.    struct node *temp = *head, *prev = *head;
51.
52.    if (*head == NULL)
53.    {
54.        *head = (struct node *)malloc(sizeof(struct node));
55.        (*head)->a = num;
56.        (*head)->left = (*head)->right = NULL;
57.    }
58.    else
59.    {
60.        while (temp != NULL)
61.        {
62.            if (num > temp->a)
63.            {
64.                prev = temp;
65.                temp = temp->right;
66.            }
67.            else
68.            {
69.                prev = temp;
70.                temp = temp->left;
71.            }
72.        }
73.        temp = (struct node *)malloc(sizeof(struct node));
74.        temp->a = num;
75.        if (num >= prev->a)
76.        {
77.            prev->right = temp;
78.        }
79.        else
80.        {
81.            prev->left = temp;
82.        }
83.    }
84.}
85.
86.void DFS(struct node *head)
87.{
88.    if (head)
89.    {
90.        if (head->left)
91.        {
92.            DFS(head->left);
93.        }
94.        if (head->right)
95.        {
96.            DFS(head->right);
97.        }
98.        printf("%d ", head->a);
99.    }
100.}
101.
102.void delete(struct node **head)
103.{
104.    if (*head != NULL)
105.    {
106.        if ((*head)->left)
107.        {
108.            delete(&(*head)->left);
109.        }
110.        if ((*head)->right)
111.        {
112.            delete(&(*head)->right);

```

```
113.    }
114.    free(*head);
115.    }
116.}
```

```
$ cc pgm34.c
$ a.out
```

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 5

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 3

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 4

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 2

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 7

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 8

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 1

Enter element to insert: 6

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 2

2 4 3 6 8 7 5

Enter your choice:

1. Insert
2. Perform DFS Traversal
3. Exit

Choice: 3
Memory Cleared
PROGRAM TERMINATED

160. C Program to Check whether the given Integer has an Alternate Pattern

[« Prev](#)

[Next »](#)

This is a C Program to check whether the given integer has an alternate pattern.

Problem Description

This C Program checks whether the given integer has an alternate pattern.

Problem Solution

Take input from the user and checks alternate pattern as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to check whether the given integer has an alternate pattern. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Check whether the given Integer has an Alternate
 * Pattern
 */
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    int num = 0, n = 0, i = 0;
    int count = 0;
    printf("Enter the number: ");
    scanf ("%d", &num);
    n = num;
    // first lets count the number of bits
    while (n)
    {
        count++;
        n = n >> 1;
    }
    printf ("\n COUNT : %d", count);

    // now check for alternative
    for (i = 0; i <= count - 2; i++)
    {
        if (((num >> i) & 1) == ((num >> (i+2)) & 1))
        {
            continue;
        }
        else
        {
            printf ("\nFALSE : ALTERNATIVE PATTERN DOES NOT EXIST\n");
            exit (0);
        }
    }
}
```

```
    }
    printf ("\nTRUE : ALTERNATIVE PATTERN DOES EXIST\n");
    return 0;
}
```

Program Explanation

1. In this C Program, we are reading the number using ‘num’ variable. Take the input from the user in the number form.
2. Count the number of bits in the given number using while loop.
3. (num >> n) & 1 is used to convert the n’th bit in the binary number. Where n is the nth position of the bit.
4. for loop statement is used to check that any alternative pattern exists or not. If there is an alternative pattern just continue. else break the statement.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered number is a valid number.

```
$ gcc alternative.c -o alternative
$ ./alternative
```

Enter the number: 10

```
COUNT : 4
TRUE : ALTERNATIVE PATTERN DOES EXIST
```

Test case 2 – Here, the entered number is a invalid number.

advertisement

```
$ gcc alternative.c -o alternative
$ ./alternative
```

Enter the number: 15

```
COUNT : 4
FALSE : ALTERNATIVE PATTERN DOES NOT EXIST
```

161. C Program to identify missing Numbers in a given Array

[« Prev](#)

[Next »](#)

This C Program identifies missing numbers in a given array.

Here is source code of the C Program to identify missing numbers in a given array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to identify missing numbers in a given array
3. */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int n, i, j, c, t, b;
9.
10.    printf("Enter size of array : ");
11.    scanf("%d", &n);
12.    int array[n - 1]; /* array size-1 */
13.    printf("Enter elements into array : \n");
14.    for (i = 0; i < n - 1; i++)
15.        scanf("%d", &array[i]);
16.    b = array[0];
17.    for (i = 1; i < n - 1; i++)
18.        b = b ^ array[i];
19.    for (i = 2, c = 1; i <= n; i++)
20.        c = c ^ i;
21.    c = c ^ b;
22.    printf("Missing element is : %d \n", c);
23.}
```

advertisement

```
$ cc bit30.c
$ a.out
Enter size of array : 6
Enter elements into array :
1
2
3
5
6
Missing element is : 4
```

162. C Program to round Floor of integer to next Lower Power of 2

[« Prev](#)

[Next »](#)

This is a C Program to round floor of integer to next lower power of 2.

Problem Description

This C Program round floor of integer to next lower power of 2

Problem Solution

Take input from the user and performs powers operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to round floor of integer to next lower power of 2. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to round floor of integer to next lower power of 2
 */
#include <stdio.h>

int main ()
{
    int num = 0;
    printf("\nEnter the number: ");
    scanf ("%d", &num);
    num--;
    num = num | (num >> 1);
    num = num | (num >> 2);
    num = num | (num >> 4);
    num = num | (num >> 8);
    num = num | (num >> 16);
    num++;
    printf ("\n NEXT NUMBER LOWER TO THE POWER OF 2 : %d\n", num);
    return 0;
}
```

Program Explanation

1. In this C Program, We are reading a number using ‘num’ variable.
2. Take the input number from the user that number will store in num variable. First subtract the 1 from the num.
3. Next we need to perform all the bits in the num set.

For Example:

advertisement

```
num = num | (num >> 1); // num = 10000 | 01000
```

```
num = num | (num >> 2); // num = 11000 | 00110
```

```
num = num | (num >> 4); // num = 11110 | 00001  
  
num = num | (num >> 8); // num = 11111 | 00000  
  
num = num | (num >> 16); // num = 11110 | 00000
```

4. After completing all shifts num will have the value that is floor to the lower power of 2.

Runtime Test Cases

Test case 1 – Enter the number which is less than 16.

```
$ gcc next_pow.c -o next_pow  
$ ./next_pow
```

```
Enter the number: 10  
NEXT NUMBER LOWER TO THE POWER OF 2 : 16
```

Test case 2 – Enter the number which is less than 5.

advertisement

```
$ gcc next_pow.c -o next_pow  
$ ./next_pow
```

```
Enter the number: 4  
NEXT NUMBER LOWER TO THE POWER OF 2 : 4
```

Test case 3 – Enter the number which is less than 10.

advertisement

```
$ gcc next_pow.c -o next_pow  
$ ./next_pow
```

```
Enter the number: 6  
NEXT NUMBER LOWER TO THE POWER OF 2 : 8
```

163. C Program To Count the Occurrence of a Substring in String

[« Prev](#)

[Next »](#)

This is a C Program to count the occurrence of a substring in string.

Problem Description

This program takes a string and a substring as input and counts the occurrence of a substring in string.

Problem Solution

1. Take a string and a substring as input.
2. Firstly check for the substring in the string.
3. If it is present then count the number of times it is present.

advertisement

Program/Source Code

Here is source code of the C Program to count occurrence of a substring in string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program To Count the Occurrence of a Substring in String
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. char str[100], sub[100];
8. int count = 0, count1 = 0;
9.
10. void main()
11. {
12.     int i, j, l, l1, l2;
13.
14.     printf("\nEnter a string : ");
15.     scanf("%[^\\n]s", str);
16.
17.     l1 = strlen(str);
18.
19.     printf("\nEnter a substring : ");
20.     scanf(" %[^\n]s", sub);
21.
22.     l2 = strlen(sub);
23.
24.     for (i = 0; i < l1;)
25.     {
26.         j = 0;
27.         count = 0;
28.         while ((str[i] == sub[j]))
29.         {
30.             count++;
31.             i++;
32.             j++;
33.         }
34.     }
35. }
```

```
34. if (count == l2)
35. {
36.     count1++;
37.     count = 0;
38. }
39. else
40.     i++;
41. }
42. printf("%s occurs %d times in %s", sub, count1, str);
43.
```

Program Explanation

1. Take a string and a substring as input and store it in the array str and sub respectively.
2. Find the length of both the strings using strlen function.
3. Using for loop find whether the substring is present or not. If it is present then count the number of times it is present using the variable count.
4. Print the variable count as output.

advertisement

Runtime Test Cases

```
Enter a string : prrrogram c prrrogramming
```

```
Enter a substring : rr
rr occurs 2 times in prrrogram c prrrogramming
```

164. C Program to Find Sum of Numbers given in Command Line Arguments Recursively

[« Prev](#)

[Next »](#)

This is a C Program to find sum of numbers given in command line arguments recursively.

Problem Description

This C Program find sum of numbers given in command line arguments recursively.

Problem Solution

This C Program Prints the sum of numbers given in command line arguments.

advertisement

Program/Source Code

Here is source code of the C Program to find sum of numbers given in command line arguments recursively. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find Sum of Numbers given in Command Line Arguments
 * Recursively
 */
#include <stdio.h>

int count, s = 0;
void sum(int *, int *);

void main(int argc, char *argv[])
{
    int i, ar[argc];
    count = argc;
    for (i = 1; i < argc; i++)
    {
        ar[i - 1] = atoi(argv[i]);
    }
    sum(ar, ar + 1);
    printf("%d", s);
}

/* computes sum of two numbers recursively */
void sum(int *a, int * b)
{
    if (count == 1)
        return;
    s = s + *a + *b;
    count -= 2;
    sum(a + 2, b + 2);
}
```

Program Explanation

In this C Program, the sum() function is used to compute the sum of two numbers recursively. If condition statement is used to check the value of ‘count’ variable is equal to 1.

advertisement

If the condition is true, then we can’t compute the summation of two numbers hence return the value. Otherwise, if the condition is false, then compute the summation of the value of ‘s’ variable with the pointer variables ‘a’ and ‘b’, and decrement the value of ‘count’ variable by 2.

Runtime Test Cases

```
$ cc arg4.c  
$ a.out 1 2 3 4  
sum is 10
```

165. C Program to Remove given Word from a String

[« Prev](#)

[Next »](#)

This is a C Program to remove given word from a string.

Problem Description

This program takes string and its substring as input and removes the substring from the string.

Problem Solution

1. Take a string and its substring as input.
2. Put each word of the input string into the rows of 2-D array.
3. Search for the substring in the rows of 2-D array.
4. When the substring is got, then override the current row with next row and so on upto the last row.

advertisement

Program/Source Code

Here is source code of the C Program to remove given word from a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. /*
3.  * C Program to Remove given Word from a String
4.  */
5. #include <stdio.h>
6. #include <stdlib.h>
7. #include <string.h>
8.
9. int main ()
10. {
11.     char str[100], word[100], twoD[10][30];
12.     int i = 0, j = 0, k = 0, len1 = 0, len2 = 0;
13.
14.     printf ("Enter the string:\n");
15.     gets (str);
16.
17.     printf ("Enter the word to be removed:\n");
18.     gets (word);
19.
20. // let us convert the string into 2D array
21. for (i = 0; str[i] != '\0'; i++)
22. {
23.     if (str[i] == ' ')
24.     {
25.         twoD[k][j] = '\0';
26.         k++;
27.         j = 0;
28.     }
29.     else
30.     {
31.         twoD[k][j] = str[i];
```

```

32.           j++;
33.       }
34.   }
35.
36. twoD[k][j] = '\0';
37.
38. j = 0;
39. for (i = 0; i < k + 1; i++)
40. {
41.     if (strcmp(twoD[i], word) == 0)
42.     {
43.         twoD[i][j] = '\0';
44.     }
45. }
46.
47. j = 0;
48.
49. for (i = 0; i < k + 1; i++)
50. {
51.     if (twoD[i][j] == '\0')
52.         continue;
53.     else
54.         printf ("%s ", twoD[i]);
55. }
56.
57. printf ("\n");
58.
59. return 0;
60.

```

Program Explanation

1. In this C program, first step is to declare the str, word and twoD arrays. str is used to store string input, word is used to store the input word and twoD is used to store each word in 2D array.
2. Next step is to take the input from the user and also take input word that has to be removed.
3. In for loop statement, traverse until the end of the string and append the character to the n th row until you encounter a white space. If a whitespace is encountered then append \0 to the current row and iterate the steps. By the end of iterations we shall have a 2D array of words. Append \0 to the end of last row.
4. Compare the words that match the given word and make them null by using strcmp function. strcmp function is used to match two strings.
5. Print the words that are not NULL in the twoD array as output and exit.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered string is a sentence with some repeated words.

```
$ gcc remove-word.c -o remove-word
$ ./remove-word
```

```
Enter the string:
Hello World hello world Hello W
Enter the word to be removed: Hello
```

```
World hello world W
```

Test case 2 – Here, the entered string is a sentence with text and numbers.

advertisement

```
$ gcc remove-word.c -o remove-word
$ ./remove-word
```

Enter the string:
simple world \0\1 0000
Enter the word to be removed: \0\1

simple world 0000

Test case 3 – Here, the entered string is a sentence with same repeated words.

```
$ gcc remove-word.c -o remove-word  
$ ./remove-word
```

Enter the string:
Sanfoundry C, Sanfoundry C, Sanfoundry C
Enter the word to be removed: Sanfoundry

C, C, C

166. C Program that uses Function to return MSB position of unsigned Integer

[« Prev](#)

[Next »](#)

This C Program uses function to return the MSB position of unsigned integer.

Here is source code of the C Program use function to return the MSB position of unsigned integer. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program that uses Function to return MSB position of unsigned Integer
3. */
4. #include <stdio.h>
5. #define NUM_BITS_INT 32
6. int int_msb_position(int n);
7.
8. void main()
9. {
10.    int n, pos;
11.
12.    printf("Enter a number : ");
13.    scanf("%d", &n);
14.    pos = int_msb_position(n);
15.    printf("\nPosition of MSB bit = %d\n", NUM_BITS_INT - (pos + 1));
16.}
17.
18./* Function to find the MSB bit position */
19.int int_msb_position(int n)
20.{
21.    int i = 0, bit;
22.    while (i < NUM_BITS_INT)
23.    {
24.        bit = n & 0x80000000;
25.        if (bit == -0x80000000)
26.        {
27.            bit = 1;
28.        }
29.        if (bit == 1)
30.            break;
31.        n = n << 1;
32.        i++;
33.    }
34.    return i;
35.}
```

advertisement

```
$ cc bit24.c
$ a.out
```

Enter a number : 127

Position of MSB bit = 6

Enter a number : 259

Position of MSB bit = 8

Enter a number : 5

Position of MSB bit = 2

167. C Program to use Bitwise Operations to Round(floor of) an Integer to next Lower Multiple of 2

[« Prev](#)

[Next »](#)

This C Program uses bitwise operations to round(floor of) an integer to next lower multiple of 2.

Here is source code of the C Program to use bitwise operations to round(floor of) an integer to next lower multiple of 2. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to use Bitwise Operations to Round(floor of) an Integer
3. * to next Lower Multiple of 2
4. */
5. #include <stdio.h>
6.
7. void main()
8. {
9.     int x = 1, i, n;
10.
11.    printf("enter the number :");
12.    scanf("%d", &n);
13.    /*for positive values */
14.    if(n > 0)
15.    {
16.        for (; x <= n >> 1;)
17.        {
18.            x = x << 1;
19.        }
20.        n = x;
21.    }
22.    /*for negative values */
23.    else
24.    {
25.        n = ~n;
26.        n = n + 1;
27.        for (; x <= n >> 1;)
28.        {
29.            x = x << 1;
30.        }
31.        x = x << 1;
32.        x = ~x;
33.        x = x + 1;
34.        n = x;
35.    }
36.    printf("%d", n);
37.}
```

advertisement

```
$ cc bit10.c
$ a.out
enter the number :9
8
$ a.out
enter the number :44
32
$ a.out
enter the number :-20
```

-32

\$ a.out

enter the number :-84

-128

168. C Program to Find Highest Frequency Character in a String

[« Prev](#)

[Next »](#)

This C Program finds highest frequency character in a string. Here this program checks which character has occurred more number of times and checks how many times these character has occurred.

Here is source code of the C Program to find the highest frequency character in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program To Find the Highest Frequency Character in a String
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. char string1[100], visited[100];
8. int count[100] = {0}, flag = 0;
9.
10. void main()
11. {
12.     int i, j = 0, k = 0, l, max, index;
13.
14.     printf("Enter a string : ");
15.     scanf("%[^\\n]s", string1);
16.
17.     l = strlen(string1);
18.
19.     for (i = 0; i < l; i++)
20.     {
21.         if (i == 0)
22.         {
23.             visited[j++] = string1[i];
24.             count[j - 1]++;
25.         }
26.         else
27.         {
28.             for (k = 0; k < j; k++)
29.             {
30.                 if (string1[i] == visited[k])
31.                 {
32.                     count[k]++;
33.                     flag = 1;
34.                 }
35.             }
36.             if (flag == 0)
37.             {
38.                 visited[j++] = string1[i];
39.                 count[j - 1]++;
40.             }
41.             flag = 0;
42.         }
43.     }
44.
45.     for (i = 0; i < j; i++)
46.     {
47.         if ((i == 0) && (visited[i] != ''))
48.         {
49.             max = count[i];
```

```
50.     continue;
51. }
52. if ((max < count[i]) && (visited[i] != ''))
53. {
54.     max = count[i];
55.     index = i;
56. }
57. }
58.
59. printf("\nMax repeated character in the string = %c ", visited[index]);
60. printf("\nIt occurs %d times", count[index]);
61. }
```

advertisement

```
$ cc string23.c
$ a.out
Enter a string : Welcome to Sanfoundry's C Programming Class !
```

```
Max repeated character in the string = o
It occurs 4 times
```

169.C Program to Display every possible Combination of two Words or Strings from the input Strings without Repeated Combinations

[« Prev](#)

[Next »](#)

This C Program displays every possible combination of two words or strings from the input strings without repeated combinations.

Here is source code of the C Program to display every possible combination of two words or strings from the input strings without repeated combinations. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Display every possible Combination of two Words
3. * or Strings from the input Strings without Repeated Combinations
4. */
5. #include <stdio.h>
6. #include <string.h>
7.
8. void main()
9. {
10. int i, j = 0, k, k1 = 0, k2 = 0, row = 0;
11. char temp[50];
12. char str1[100], str2[100], str1[5][20], str3[6][20], str4[60][40];
13.
14. printf("enter the string :");
15. scanf(" %[^\n]s", &str);
16. printf("enter string:");
17. scanf(" %[^\n]s", &str2);
18.
19./* read strings into 2d character arrays */
20. for (i = 0; str[i] != '\0'; i++)
21. {
22.     if (str[i] == ' ')
23.     {
24.         str1[k1][j] = '\0';
25.         k1++;
26.         j = 0;
27.     }
28.     else
29.     {
30.         str1[k1][j] = str[i];
31.         j++;
32.     }
33. }
34. str1[k1][j] = '\0';
35. j = 0;
36. for (i = 0; str2[i] != '\0'; i++)
37. {
38.     if (str2[i] == ' ')
39.     {
40.         str3[k2][j] = '\0';
41.         k2++;
42.         j = 0;
43.     }
44.     else
45.     {
46.         str3[k2][j] = str2[i];
```

```

47.         j++;
48.     }
49. }
50. str3[k2][j] = '\0';
51.
52./* concatenates string1 words with string2 and stores in 2d array */
53. row = 0;
54. for (i = 0;i <= k1;i++)
55. {
56.     for (j = 0;j <= k2;j++)
57.     {
58.         strcpy(temp, str1[i]);
59.         strcat(temp, str3[j]);
60.         strcpy(str4[row], temp);
61.         row++;
62.     }
63. }
64. for (i = 0;i <= k2;i++)
65. {
66.     for (j = 0;j <= k1;j++)
67.     {
68.         strcpy(temp, str3[i]);
69.         strcat(temp, str1[j]);
70.         strcpy(str4[row], temp);
71.         row++;
72.     }
73. }
74.
75./* eliminates repeated combinations */
76. for (i = 0;i < row;i++)
77. {
78.     for (j = i + 1;j < row;j++)
79.     {
80.         if (strcmp(str4[i], str4[j]) == 0)
81.         {
82.             for (k = j;k <= row;k++)
83.             {
84.                 strcpy(str4[k], str4[k + 1]);
85.             }
86.             row--;
87.         }
88.     }
89. }
90.
91./* displays the output */
92. for (i = 0;i < row;i++)
93. {
94.     printf("\n%s", str4[i]);
95. }
96.}

```

advertisement

```

$ cc string27.c
$ a.out
enter the string :welcome to sanfoundry's class
enter string:welcome to c programming class

```

```

welcomewelcome
welcometo
welcomec
welcomeprogramming
welcomeclass
towelcome
toto
toc
toprogramming

```

to class
sanfoundry's welcome
sanfoundry's to
sanfoundry's sc
sanfoundry's programming
sanfoundry's class
class welcome
class to
class sc
class programming
class class
welcome sanfoundry's
to sanfoundry's
c welcome
cto
csanfoundry's
c class
programming welcome
programming to
programming sanfoundry's
programming class
class sanfoundry's

170.C program to Display the Function Names defined in C Source File

[« Prev](#)

[Next »](#)

This C Program displays the function names defined in c source file.

Here is source code of the C Program to display the function names defined in c source file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C program to Display the Function Names defined in C Source File
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. void check(char *c,int p1, int p2);
8. void display(char *c, int p1);
9.
10.void main(int argc, char **argv)
11{
12    FILE *fp;
13    char ch[100];
14    char *pos1, *pos2, *pos3;
15.
16    fp=fopen(argv[1], "r");
17    if (fp == NULL)
18    {
19        printf("\nFile unable to open");
20        return;
21    }
22    else
23        printf("\nFile Opened to display function names \n");
24    while (1)
25    {
26        if ((fgets(ch, 100, fp)) != NULL)
27        {
28            if ((strstr(ch, "/*")) == NULL)
29            {
30                pos1 = strchr(ch, '(');           /* check opening brace */
31                if (pos1)
32                {
33                    pos2 = strchr(ch, ')');       /* check oclosing brace */
34                    if (pos2)
35                    {
36                        pos3 = strchr(ch, ';');   /* check for semicolon */
37                        if ((pos1 < pos2) && (pos3 == NULL) || (pos3 < pos1))
38                        {
39                            check(ch, pos1 - ch, pos2 - ch);
40                        }
41                        else continue;
42                    }
43                    else continue;
44                }
45                else continue;
46            }
47            else continue;
48        }
```

```

49.     else break;
50. }
51. fclose(fp);
52. }
53.
54./* To check if it is a function */
55.void check(char *c, int p1, int p2)
56.{
57.    int i, flag = 0, temp = p1;
58.
59.    if ((c[p1 + 1] == ')'))
60.    {
61.        display(c, p1);
62.        return;
63.    }
64.    for (i = p1 + 1; i < p2; i++)
65.    {
66.        if ((c[i] != ')') || (c[i] == ')'))
67.        {
68.            flag = 1;
69.
70.        }
71.        if (flag == 0)
72.        {
73.            display(c, p1);
74.            return;
75.        }
76.        else
77.        {
78.            flag = 0;
79.            while (c[--temp] != ')');
80.            for (i = 0; i < temp; i++)
81.                if (c[i]==')')
82.                {
83.                    flag = 1;
84.                }
85.            if (flag == 0)
86.            {
87.                display(c, p1);
88.                return;
89.            }
90.            else
91.                return;
92.        }
93.    }
94.}
95.
96./* To display function name */
97.void display(char *c,int p1)
98.{
99.    int temp = p1, i;
100.
101.    while (c[--temp] != ')');
102.    for (i = temp + 1; i < p1; i++) /* Print name of function character by character */
103.        printf("%c", c[i]);
104.    printf("\n");
105.    return;
106.
107.}

```

advertisement

```

$ cc file9.c
$ a.out

```

File Opened to display **function** names :
main

171.C Program to Reverse every Word of given String

[« Prev](#)

[Next »](#)

This is a C Program to reverse every word of given string.

Problem Description

This program takes a string and reverses every word of the string.

Problem Solution

1. Take a string as input.
2. Store each word of the input string in the separate rows of the 2-D array.
3. Reverse each word of the string.

advertisement

Program/Source Code

Here is source code of the C Program to reverse every word of given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2. /*  
3.  * C Program to Reverse every Word of given String  
4.  */  
5. #include <stdio.h>  
6. #include <string.h>  
7.  
8. void main()  
9. {  
10.    int i, j = 0, k = 0, x, len;  
11.    char str[100], str1[10][20], temp;  
12.  
13.    printf("enter the string :");  
14.    scanf("%[^\\n]s", str);  
15.  
16./* reads into 2d character array */  
17.    for (i = 0; str[i] != '\0'; i++)  
18.    {  
19.        if (str[i] == ' ')  
20.        {  
21.            str1[k][j] = '\0';  
22.            k++;  
23.            j = 0;  
24.        }  
25.        else  
26.        {  
27.            str1[k][j] = str[i];  
28.            j++;  
29.        }  
30.    }  
31.    str1[k][j] = '\0';  
32.
```

```
33./* reverses each word of a given string */
34. for (i = 0;i <= k;i++)
35. {
36.     len = strlen(str1[i]);
37.     for (j = 0, x = len - 1;j < x;j++,x--)
38.     {
39.         temp = str1[i][j];
40.         str1[i][j] = str1[i][x];
41.         str1[i][x] = temp;
42.     }
43. }
44. for (i = 0;i <= k;i++)
45. {
46.     printf("%s ", str1[i]);
47. }
48.}
```

Program Explanation

1. Take a string as input and store it in the array str[].
2. Using for loop store each word of the input string into the 2-D array str1[][].
3. In the 2-D array str1[][] reverse each word of the string at each row of the array.
4. Print the 2-D array as output.

advertisement

Runtime Test Cases

enter the string :C Programming Class
C gnimmarginP ssalC

172.C Program to Find the Size of File using File Handling Function

[« Prev](#)

[Next »](#)

This C Program finds the size of file using file handling function.

Here is source code of the C Program to find the size of file using file handling function. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Size of File using File Handling Function
3. */
4. #include <stdio.h>
5.
6. void main(int argc, char **argv)
7. {
8.     FILE *fp;
9.     char ch;
10.    int size = 0;
11.
12.    fp = fopen(argv[1], "r");
13.    if (fp == NULL)
14.        printf("\nFile unable to open ");
15.    else
16.        printf("\nFile opened ");
17.    fseek(fp, 0, 2); /*file pointer at the end of file */
18.    size = ftell(fp); /*take a position of file pointer un size variable */
19.    printf("The size of given file is : %d\n", size);
20.    fclose(fp);
21.}
```

advertisement

```
$ vi file10.c
$ cc file10.c
$ a.out myvmlinux
```

```
File opened The size of given file is : 3791744
```

173.C Program to List All Lines containing a given String

[« Prev](#)

[Next »](#)

This C Program listS all lines containing a given string.

Here is source code of the C Program to list all lines containing a given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to List All Lines containing a given String
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <string.h>
7.
8. int search(FILE *, char *);
9.
10. void main(int argc, char * argv[])
11. {
12.     FILE *fp1;
13.     int p;
14.
15.     fp1 = fopen(argv[1], "r+");
16.     if(fp1 == NULL)
17.     {
18.         printf("cannot open the file ");
19.         exit(0);
20.     }
21.     search(fp1, argv[2]);
22.     fclose(fp1);
23. }
24.
25./* Searches the lines */
26. int search(FILE *fp, char * str)
27. {
28.     FILE *fp1;
29.     fp1 = fopen("fp1","w");
30.     char s[10],c;
31.     int len = strlen(str);
32.     int i = 0;
33.     int d;
34.     int seek = fseek(fp, 0, 0);
35.     c = fgetc(fp);
36.     while (c != EOF)
37.     {
38.         if (c == ' ' || c == '\n')
39.         {
40.             s[i] = '\0';
41.             i = 0;
42.             if (strcmp(s, str) == 0)
43.             {
44.                 while (c = fgetc(fp) != '\n')
45.                 {
46.                     fseek(fp, -2L, 1);
```

```
47.         d = ftell(fp);
48.     }
49.     while ((c = fgetc(fp)) != '\n')
50.     {
51.         fputc(c, fp1);
52.     }
53. }
54. }
55. else
56. {
57.     s[i] = c;
58.     i++;
59. }
60. c = fgetc(fp);
61. }
62. return 1;
63. }
```

advertisement

```
$cat example
hi hello everyone
again hi to the late comers
welcome to the class
```

```
$ cc file6.c
$ ./a.out example
hi hello everyone
again hi to the late comers
```

174.C Program to Print the Range of Fundamental Data Types

[« Prev](#)

[Next »](#)

This is a C Program to print the range of data types.

Problem Description

This program prints the range of data types.

Problem Solution

1. Convert the bytes into bits.
2. For signed data types, use formula $-2^{(n-1)}$ to $(2^{(n-1)})-1$.
3. For unsigned data types, use formula $(2^n) - 1$. Where n is the number of bits in both the cases.

advertisement

Program/Source Code

Here is source code of the C Program to print the range. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Print the Range
3. */
4. #include <stdio.h>
5. #define SIZE(x) sizeof(x)*8
6.
7. void signed_one(int);
8. void unsigned_one(int);
9.
10.void main()
11.{  
12.    printf("\nrange of int");
13.    signed_one(SIZE(int));
14.    printf("\nrange of unsigned int");
15.    unsigned_one(SIZE(unsigned int));
16.    printf("\nrange of char");
17.    signed_one(SIZE(char));
18.    printf("\nrange of unsigned char");
19.    unsigned_one(SIZE(unsigned char));
20.    printf("\nrange of short");
21.    signed_one(SIZE(short));
22.    printf("\nrange of unsigned short");
23.    unsigned_one(SIZE(unsigned short));
24.
25.}
26./* RETURNS THE RANGE SIGNED*/
27.void signed_one(int count)
28.{
```

```

29. int min, max, pro;
30. pro = 1;
31. while (count != 1)
32. {
33.     pro = pro << 1;
34.     count--;
35. }
36. min = ~pro;
37. min = min + 1;
38. max = pro - 1;
39. printf("\n%d to %d", min, max);
40.
41./* RETURNS THE RANGE UNSIGNED */
42.void unsigned_one(int count)
43.{
44.    unsigned int min, max, pro = 1;
45.
46.    while (count != 0)
47.    {
48.        pro = pro << 1;
49.        count--;
50.    }
51.    min = 0;
52.    max = pro - 1;
53.    printf("\n%u to %u", min, max);
54.

```

Program Explanation

1. Convert the number of bytes into bits by multiplying the bytes with 8.
2. Use two functions namely signed_one() and unsigned_one() for calculating the range of signed and unsigned data types respectively.
3. Value got at step 1 is sent as a parameter to both the functions. In both the function, it is received by variable count.
4. Initialize the variable pro to 1 in both the functions.
5. In the function signed_one() using while loop with the condition (count != 1), shift the variable pro to its left by 1 position and decrement the variable count by 1 consecutively.
6. When the loop terminates, assign the complement of pro to the variable min and increment the min by 1. Decrement the variable pro and assign it to the variable max. Print min and max as output.
7. In the function unsigned_one() using while loop with the condition (count != 0), shift the variable pro to its left by 1 position and decrement the variable count by 1 consecutively.
8. When the loop terminates, assign zero to the variable min. Decrement the variable pro and assign it to the variable max. Print min and max as output.

advertisement

Runtime Test Cases

```

range of int
-2147483648 to 2147483647
range of unsigned int
0 to 4294967295
range of char
-128 to 127
range of unsigned char
0 to 255
range of short
-32768 to 32767
range of unsigned short
0 to 65535

```

175.C Program to Concatenate two Strings Lexically

[« Prev](#)

[Next »](#)

This is a C program to concatenate two strings lexically.

Problem Description

This C Program concatenates the given two strings lexically.

Problem Solution

Take input from the user and concatenates the two strings as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to concatenate the given two strings lexically. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Concatenate the given two Strings Lexically
 */
#include <string.h>
#include <stdio.h>

void sort(char *p);

void main()
{
    char string1[100], string2[100];
    int i, len, j;

    printf("\nEnter a string : ");
    scanf("%[^\\n]s", string1);
    printf("\nEnter another string to concat : ");
    scanf(" %[^\\n]s", string2);
    len = strlen(string1);
    string1[len] = '\0';
    for(i = 0, j = len + 1; i < strlen(string2); i++, j++)
        string1[j] = string2[i];
    string1[j] = '\0';
    sort(string1);
}

/* Sorting to make concatenation lexical */
```

```

void sort(char *p)
{
    char temp[100];
    char a[100][100];
    int t1, i, j = 0, k = 0, l = strlen(p), x = 0, y = 0, z = 0, count, l1, l2;

    for (i = 0; i < l; i++)
    {
        if (p[i] != ' ')
        {
            a[k][j++] = p[i];
        }
        else
        {
            a[k][j] = '\0';
            k++;
            j = 0;
        }
    }

    t1 = k;
    k = 0;

    for (i = 0; i < t1; i++)
    {
        for (j = i + 1; j <= t1; j++)
        {
            l1 = strlen(a[i]);
            l2 = strlen(a[j]);
            if (l1 > l2)
                count = l1;
            else
                count = l2;
            x = 0, y = 0;
            while ((x < count) || (y < count))
            {
                if (a[i][x] == a[j][y])
                {
                    x++;
                    y++;
                    continue;
                }
                else
                    if (a[i][x] < a[j][y]) break;
                else
                    if (a[i][x] > a[j][y])
                {
                    for (z = 0; z < l2; z++)
                    {
                        temp[z] = a[j][z];
                        a[j][z] = '\0';
                    }
                    temp[z] = '\0';

                    for (z = 0; z < l1; z++)
                    {
                        a[j][z] = a[i][z];
                        a[i][z] = '\0';
                    }
                    a[j][z] = '\0';

                    for (z = 0; z < strlen(temp); z++)
                    {
                        a[i][z] = temp[z];
                    }
                    break;
                }
            }
        }
    }
}

```

```

        }

    }

}

for (i = 0; i < l; i++)
p[i] = '\0';
k = 0;
j = 0;
for (i = 0; i < l; i++)
{
    if (a[k][j] != '\0')
    {
        p[i] = a[k][j++];
    }
    else
    {
        k++;
        j = 0;
        p[i] = '\0';
    }
}
puts(p);
}

```

Program Explanation

In this C program, We are reading two strings using the string1[] and string2[] array variables. The memset() function is used to initialize the value of string to NULL.

advertisement

Using for loop simply traverse the value of ‘string1’ variable and assign the value of ‘i’ variable to ‘pos’ variable. Concatenate the second string to the end of the first string.

Another for loop is used to assign the value of ‘string2[]’ array character variable to ‘string1[]’ array variable. Assign the last character of string1 to null and print the concatenated value of the strings.

Runtime Test Cases

```
$ cc string17.c
$ a.out
```

Enter a string : hello this

Enter another string to concat : is sanfoundry
hello is sanfoundry this

176.C Program to Print the Words Ending with Letter S

[« Prev](#)

[Next »](#)

This is a C Program to print the words ending with letter s.

Problem Description

This program takes a string as input and print the words ending with letter s.

Problem Solution

1. Take a string as input.
2. Find the blank space in the string and then check if the previous letter is ‘s’ or not.
3. If it is ‘s’ then print the corresponding word.

advertisement

Program/Source Code

Here is source code of the C Program to print the words ending with letter s. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.
2. /*
3.  * C Program to Print the Words Ending with Letter S
4. */
5. #include <stdio.h>
6. #include <string.h>
7.
8. char str[100];
9.
10.void main()
11.{{
12.    int i, t, j, len;
13.
14.    printf("Enter a string : ");
15.    scanf("%[^\\n]s", str);
16.
17.    len = strlen(str);
18.
19.    str[len] = '\0';
20.
21.    for (t = 0, i = 0; i < strlen(str); i++)
22.    {
23.        if ((str[i] == ' ') && (str[i - 1] == 's'))
24.        {
```

```
25.     for (j = t; j < i; j++)
26.         printf("%c", str[j]);
27.     t = i + 1;
28.     printf("\n");
29. }
30. else
31. {
32.     if(str[i] == ' ')
33.     {
34.         t = i + 1;
35.     }
36. }
37. }
38. }
```

Program Explanation

1. Take a string as input and store it in the array str[].
2. Find its length and add a blank space to the input string at last.
3. Using for loop find the blank space in the string and check whether its previous letter is 's' or not.
4. If it is 's' then print the corresponding word. Otherwise repeat the steps 3-4 until the last element of the string.

advertisement

Runtime Test Cases

```
Enter a string : Welcome to Sanfoundry's C Programming Class, Welcome Again to C Class !
Sanfoundry's
Class
```

177.C Program to Find the Sum of ASCII values of All Characters in a given String

[« Prev](#)

[Next »](#)

This is a C Program to find the sum of ASCII values of all characters in a given string.

Problem Description

This program takes a string as input and finds the sum of ASCII values of all characters in a given string.

Problem Solution

1. Take a string as input and store it in the array.
2. Add all the elements of the array.
3. Print the output and exit.

advertisement

Program/Source Code

Here is source code of the C Program to find the sum of ASCII values of all characters in a given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1.  
2. /*  
3.  * C Program To Find the Sum of ASCII values of All Characters in a  
4.  * given String  
5.  */  
6. #include <stdio.h>  
7. #include <string.h>  
8.  
9. void main()  
10.{  
11. int sum = 0, i, len;  
12. char string1[100];  
13.  
14. printf("Enter the string : ");  
15. scanf("%[^\\n]s", string1);  
16. len = strlen(string1);  
17. for (i = 0; i < len; i++)  
18. {  
19.     sum = sum + string1[i];  
20. }  
21. printf("\nSum of all characters : %d ",sum);  
22. }
```

Program Explanation

1. Take a string as input and store it in the array string[].
2. Initialize the variable sum to zero. Using for loop increment the variable sum with the elements of the array.
3. Print the variable sum as output.

advertisement

Runtime Test Cases

Enter the string : Welcome to Sanfoundry's C Programming Class, Welcome Again to C Class !

Sum of all characters : 6307

178.C Program to Print Environment Variables

« [Prev](#)

[Next](#) »

This C Program prints environment variables. Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer.

Here is source code of the C Program to print environment variables. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Print Environment variables
3. */
4. #include <stdio.h>
5.
6. void main(int argc, char *argv[], char * envp[])
7. {
8.     int i;
9.
10.    for (i = 0; envp[i] != NULL; i++)
11.    {
12.        printf("\n%s", envp[i]);
13.    }
14.}
```

advertisement

```
$ cc arg7.c
$ a.out
```

```
HOSTNAME=localhost.localdomain
SELINUX_ROLE_REQUESTED=
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
SSH_CLIENT=192.168.7.43 49162 22
SELINUX_USE_CURRENT_RANGE=
QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
SSH_TTY=/dev/pts/8
USER=harika
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd=40;33:01:or=40;31:01:mi=01;05;37
;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh
=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*
.bz2=01;31:*.tbz=01;31:*.tbz2=01;31:*.bz=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.rar=01;31:*.ace=01;31:*.z
oo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:
*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01
;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35
:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.ASF=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01
;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;3
5:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=01;36:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=
01;36:*.mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;36:*.oga=01;36:*.spx=01;36:*.xspf=01;36:
PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/harika/bin:
MAIL=/var/spool/mail/harika
PWD=/home/harika
KDE_IS_PRELINKED=1
LANG=en_US.UTF-8
KDEDIRS=/usr
```

```
SELINUX_LEVEL_REQUESTED=
HISTCONTROL=ignoredups
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HOME=/home/harika
SHLVL=2
LOGNAME=harika
CVS_RSH=ssh
QTLIB=/usr/lib64/qt-3.3/lib
SSH_CONNECTION=192.168.7.43 49162 192.168.7.140 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILERAMES=1
./a.out
OLDPWD=/home
```

179.C Program to Copy File into Another File

[« Prev](#)

[Next »](#)

This C Program copies a file into another file.

Here is source code of the C Program to copy a file into another file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Copy a File into Another File
3. */
4. #include <stdio.h>
5.
6. void main(int argc,char **argv)
7. {
8.     FILE *fp1, *fp2;
9.     char ch;
10.    int pos;
11.
12.    if ((fp1 = fopen(argv[1],"r")) == NULL)
13.    {
14.        printf("\nFile cannot be opened");
15.        return;
16.    }
17.    else
18.    {
19.        printf("\nFile opened for copy...\n");
20.    }
21.    fp2 = fopen(argv[2], "w");
22.    fseek(fp1, 0L, SEEK_END); //file pointer at end of file
23.    pos = ftell(fp1);
24.    fseek(fp1, 0L, SEEK_SET); //file pointer set at start
25.    while (pos--)
26.    {
27.        ch = fgetc(fp1); // copying file character by character
28.        fputc(ch, fp2);
29.    }
30.    fcloseall();
31.}
```

advertisement

```
$ cc file1.c
$ a.out /tmp/vmlinux mylinux
```

File opened for copy...

```
$ cmp /tmp/vmlinux mylinux
```

```
$ ls -l mylinux
-rw-rw-r--. 1 adi adi 3791744 Jul 27 19:57 mylinux
```

180.C Program to find Next higher Value of N with same 1's

[« Prev](#)

[Next »](#)

This is a C Program to find next higher value of n with same 1's.

Problem Description

This C Program next higher value of n with same 1's.

Problem Solution

Take input from the user and perform bit operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to next higher value of n with same 1's. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to next higher value of n with same 1's
 */
#define NUM_BITS_INT 32
#include <stdio.h>
#include <stdlib.h>
int newcount(int);
void main()
{
    int count1 = 0, k = 0, j, t, n, bit, i = 1, count = 0;
    printf("Enter a number : ");
    scanf("%d", &n);
    t = n;

    if (t == 0)
    {
        printf ("\nThe next highest number is : 1\n");
        exit (0);
    }

    while(t != 0)
    {
        bit = t & 0x80000000;
        if (bit == -0x80000000)
        {
            bit = 1;
        }
        if (bit == 1)
            count++;
        t = t << 1;

    }
    for (k = n + 1;k++)
}
```

```

{
    count1 = newcount(k);
    if(count1 == count)
    {
        printf("The next highest number is : %d ", k);
        break;
    }
}

/* To count the no. of 1's in the no. */
int newcount(int k)
{
    int bit, count = 0;
    while (k != 0)
    {
        bit = k & 0x80000000;
        if(bit == -0x80000000)
        {
            bit = 1;
        }
        if(bit == 1)
            count++;
        k = k << 1;
    }
    return(count);
}

```

Program Explanation

In this C Program, we are reading the number using ‘n’ variable. While loop is used to check that the value of ‘t’ variable is not equal to 0. If the condition is true, then execute the statement.

advertisement

Binary AND operator is used to copy a bit to the ‘bit’ operator. If condition statement is used to check that the value of ‘bit’ variable is equal to -0x80000000. If the condition is true then execute the statement and assign the value of ‘bit’ variable as 1.

Another if condition statement is used to check that the value of ‘bit’ variable is equal to 1. If the condition is true, then execute the statement and increment the value of ‘count’ variable. Binary left shift operator is used for moving the number of bits value 1 to left by the number of bits specified by the value of ‘t’ variable and assign the value to ‘t’ variable.

For loop is used to print the next highest number. The newcount() function is used to count the number of 1’s in the number. While loop is used to check that the value of ‘k’ variable is not equal to 0. If the condition is true, then execute the statement. Binary AND operator is used to copy a bit to the ‘bit’ operator.

advertisement

If condition statement is used to check that ‘bit’ variable value is equal to -0x80000000. If the condition is true then execute the statement and assign the ‘bit’ variable value as 1. Another if condition statement is used to check that the value of ‘bit’ variable is equal to 1. If the condition is true, then execute the statement and increment the value of count variable.

Binary left shift operator is used for moving the number of bits 1 value to left by the number of bits specified by the ‘k’ variable value and assign the value to ‘k’ variable. If condition statement is used to check that both the values of ‘count1’ and ‘count’ variables are equal. If the condition is true then execute the statement and print the next highest value in the number with same 1’s.

Runtime Test Cases

```
$ cc bit18.c
$ a.out
Enter a number : 128
The next highest number is : 256
Enter a number : 127
The next highest number is : 191
Enter a number : 6
The next highest number is : 9
Enter a number : 12
The next highest number is : 17
```

181.C Program to Check if All the Bits of a given Integer is One(1)

[« Prev](#)

[Next »](#)

This is a C Program to check if all the bits of a given integer is one.

Problem Description

This C Program to check if all the bits of a given integer is one.

Problem Solution

Take input from the user and performs bits operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to check if all the bits of a given integer is one. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to check if all the bits of a given integer is one(1)
 */
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    int num = 0, count = 0, n = 0, i = 0;

    printf ("\nEnter the number : ");
    scanf ("%d", &num);
    n = num;
    if (num == 0)
    {
        printf ("\nFALSE : ALL BITS ARE NOT SET\n");
        exit (0);
    }
    while (n)
    {
        count++;
        n = n >> 1;
    }
    for (i = 0; i < count; i++)
    {
        if (((num >> i) & 1) == 1)
        {
            continue;
        }
        else
        {
            printf ("\nFALSE : ALL BITS ARE NOT SET\n");
        }
    }
}
```

```
        exit (0);
    }
}
printf ("\nTRUE : ALL BITS ARE SET\n");
return 0;
}
```

Program Explanation

1. In this C program, we are reading the number using ‘num’ variable. Take the input number from the user and make a copy of the number to n.
2. If the number given is zero we just print NOT SET and exit.
3. While loop is used to count the number of bits. $n = n \gg 1$; it performs right shift by 1 bit. Traverse through each bit. continue if the bit is set else print NOT SET and exit.
4. $(num \gg n) \& 1$; is used to retrieve the nth bit from the number num.
5. For loop is used to check whether all bits are set. If all bits are set then print “ALL BITS ARE SET” and return.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered number is 0.

```
$ cc bit22.c
$ a.out
Enter the number : 0
```

FALSE : ALL BITS ARE NOT SET

Test case 2 – Here, the entered number is 5.

advertisement

```
$ cc bit22.c
$ a.out
Enter the number : 5
```

FALSE : ALL BITS ARE NOT SET

Test case 3 – Here, the entered number is 7.

```
$ cc bit22.c
$ a.out
Enter the number : 7
```

TRUE : ALL BITS ARE SET

Test case 4 – Here, the entered number is 127.

advertisement

```
$ cc bit22.c
$ a.out
Enter the number : 127
```

TRUE : ALL BITS ARE SET

182.C Program to Count the Number of Trailing Zeroes in Integer

[« Prev](#)

[Next »](#)

This is a C Program to count the number of trailing zeroes in integer.

Problem Description

This C Program counts the number of trailing zeroes in integer.

Problem Solution

Take input from the user and counts the number of trailing zeroes in given integer as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to count the number of trailing zeroes in integer. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Count the Number of Trailing Zeroes in Integer
 */
#include <stdio.h>

void main()
{
    int j = 31, i, count = 0;
    unsigned int num;
    int b[32] = {0};

    printf("enter the number:");
    scanf("%d", &num);
    while (num != 0)
    {
        if (num & 1 == 1)
        {
            break;
        }
        else
        {
            count++;
            num = num >> 1;
        }
    }
    printf("\n%d", count);
}
```

Program Explanation

This C Program we are reading the number using ‘num’ variable. While condition statement is used to check that the number is not equal to 0. If the condition is true then execute the statement.

advertisement

If else condition statement is used to check that the copy of the bit 1 in the value of ‘num’ variable is equal to the value of 1. If the condition is true, then exit the condition statement using break statement.

Otherwise, if the condition is false then execute the else statement by increment the value of ‘count’ variable. Using binary right shift operator the value 1 is moved right by the number of bits specified by the value of ‘num’ variable and assign to ‘num’ variable. Print the number of trailing zero’s in integer.

Runtime Test Cases

```
$ cc bit4.c
$ ./a.out
enter the number:128
7
$ ./a.out
enter the number:-127
0
```

183.C Program to Create Employee Record and Update it

[« Prev](#)

[Next »](#)

This C Program creates employee record and update it.

Here is source code of the C Program to create employee record and update it. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Create Employee Record and Update it
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <string.h>
7. #define size 200
8.
9. struct emp
10. {
11.     int id;
12.     char *name;
13. }*emp1, *emp3;
14.
15. void display();
16. void create();
17. void update();
18.
19. FILE *fp, *fp1;
20. int count = 0;
21.
22. void main(int argc, char **argv)
23. {
24.     int i, n, ch;
25.
26.     printf("[1] Create a Record\n");
27.     printf("[2] Display Records\n");
28.     printf("[3] Update Records\n");
29.     printf("[4] Exit");
30.     while (1)
31.     {
32.         printf("\nEnter your choice : ");
33.         scanf("%d", &ch);
34.         switch (ch)
35.         {
36.             case 1:
37.                 fp = fopen(argv[1], "a");
38.                 create();
39.                 break;
40.             case 2:
41.                 fp1 = fopen(argv[1],"rb");
42.                 display();
43.                 break;
44.             case 3:
45.                 fp1 = fopen(argv[1], "r+");
46.                 update();
47.                 break;
48.             case 4:
49.                 exit(0);
50.         }
51.     }
}
```

```

52.
53.
54./* To create an employee record */
55.void create()
56.{
57.    int i;
58.    char *p;
59.
60.    emp1 = (struct emp *)malloc(sizeof(struct emp));
61.    emp1->name = (char *)malloc((size)*(sizeof(char)));
62.    printf("Enter name of employee : ");
63.    scanf(" %[^\n]s", emp1->name);
64.    printf("Enter emp id : ");
65.    scanf(" %d", &emp1->id);
66.    fwrite(&emp1->id, sizeof(emp1->id), 1, fp);
67.    fwrite(emp1->name, size, 1, fp);
68.    count++; // count to number of entries of records
69.    fclose(fp);
70.}
71.
72./* Display the records in the file */
73.void display()
74.{
75.    emp3= (struct emp *)malloc(1*sizeof(struct emp));
76.    emp3->name=(char *)malloc(size*sizeof(char));
77.    int i = 1;
78.
79.    if (fp1 == NULL)
80.        printf("\nFile not opened for reading");
81.    while (i <= count)
82.    {
83.        fread(&emp3->id, sizeof(emp3->id), 1, fp1);
84.        fread(emp3->name, size, 1, fp1);
85.        printf("\n%d %s",emp3->id,emp3->name);
86.        i++;
87.    }
88.    fclose(fp1);
89.    free(emp3->name);
90.    free(emp3);
91.}
92.
93.void update()
94.{
95.    int id, flag = 0, i = 1;
96.    char s[size];
97.
98.    if (fp1 == NULL)
99.    {
100.        printf("File cant be opened");
101.        return;
102.    }
103.    printf("Enter employee id to update : ");
104.    scanf("%d", &id);
105.    emp3 = (struct emp *)malloc(1*sizeof(struct emp));
106.    emp3->name=(char *)malloc(size*sizeof(char));
107.    while(i<=count)
108.    {
109.        fread(&emp3->id, sizeof(emp3->id), 1, fp1);
110.        fread(emp3->name,size,1,fp1);
111.        if (id == emp3->id)
112.        {
113.            printf("Enter new name of employee to update : ");
114.            scanf(" %[^\n]s", s);
115.            fseek(fp1, -204L, SEEK_CUR);
116.            fwrite(&emp3->id, sizeof(emp3->id), 1, fp1);
117.            fwrite(s, size, 1, fp1);

```

```
118.     flag = 1;
119.     break;
120.   }
121.   i++;
122. }
123. if (flag != 1)
124. {
125.   printf("No employee record found");
126.   flag = 0;
127. }
128. fclose(fp1);
129. free(emp3->name); /* to free allocated memory */
130. free(emp3);
131. }
```

advertisement

```
$ a.out emprec1
```

- 1] Create a Record
- 2] Display Records
- 3] Update Records
- 4] Exit

```
Enter your choice : 1
```

```
Enter name of employee : aaa
```

```
Enter emp id : 100
```

```
Enter your choice : 1
```

```
Enter name of employee : bbb
```

```
Enter emp id : 200
```

```
Enter your choice : 1
```

```
Enter name of employee : ccc
```

```
Enter emp id : 300
```

```
Enter your choice : 1
```

```
Enter name of employee : ddd
```

```
Enter emp id : 400
```

```
Enter your choice : 1
```

```
Enter name of employee : eee
```

```
Enter emp id : 500
```

```
Enter your choice : 2
```

```
100 aaa
```

```
200 bbb
```

```
300 ccc
```

```
400 ddd
```

```
500 eee
```

```
Enter your choice : 3
```

```
Enter employee id to update : 300
```

```
Enter new name of employee to update : cdefgh
```

```
Enter your choice : 2
```

```
100 aaa
```

```
200 bbb
```

```
300 cdefgh
```

```
400 ddd
```

```
500 eee
```

```
Enter your choice : 4
```

184.C Program to find Longer Repeating Sequence

[« Prev](#)

[Next »](#)

This C Program find Longer Repeating Sequence.

Here is source code of the C Program to find Longer Repeating Sequence. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to find Longer Repeating Sequence
3.  */
4. #include <stdio.h>
5. #include <string.h>
6.
7. void main()
8. {
9.     char s1[100], ar[10][20], ar1[10][20], new[10];
10.    int i, j = 0, k = 0, l, count = 0, flag = 0, n, temp, len[20], maxlen = 0;
11.
12.    printf("\nEnter the string:");
13.    scanf(" %[^\n]s", s1);
14.
15. /*COPYING GIVEN STRING TO 2D ARRAY*/
16.    for (i = 0; s1[i] != '\0'; i++, j++)
17.    {
18.        if (s1[i] >= 33 && s1[i] <= 64)
19.            i++;
20.        if (s1[i] == ' ')
21.        {
22.            ar[k][j] = '\0';
23.            k++;
24.            i++;
25.            j = 0;
26.        }
27.        ar[k][j] = s1[i];
28.    }
29.    ar[k][j] = '\0';
30. /*PLACING THE REPEATED WORDS AND LENGTHS INTO NEW ARRAY*/
31. l = 0;
32. for (i = 0; i <= k; i++)
33. {
34.     for (j = i + 1; j <= k; j++)
35.     {
36.         if (strcmp(ar[i], ar[j]) == 0)
37.         {
38.             for (n = 0; n < 1 && l != 0; n++)
39.             {
40.                 if (strcmp(ar[i], ar1[k]) == 0)
41.                 {
42.                     flag = 1;
43.                     break;
44.                 }
45.             }
46.             if (flag != 1)
47.             {
48.                 strcpy(ar1[l], ar[i]);
49.                 len[l] = strlen(ar1[l]);
50.                 l++;
51.             }
52.         }
53.     }
54. }
55. 
```

```

52.         flag = 0;
53.         break;
54.     }
55. }
56. printf("\n");
57. /*SORTING IS DONE BASED ON THEIR LENGTHS*/
58. for (i = 0;i < l;i++)
59. {
60.     for (j = i + 1;j < l;j++)
61.     {
62.         if (len[i] < len[j])
63.         {
64.             temp = len[i];
65.             strcpy(new, ar1[i]);
66.             len[i] = len[j];
67.             strcpy(ar1[i], ar1[j]);
68.             len[j] = temp;
69.             strcpy(ar1[j], new);
70.         }
71.     }
72. }
73. maxlen = len[0];
74. for (i = 0;i < l;i++)
75. {
76.     if (len[i] == maxlen)
77.         printf("\nThe longer repeating sequence of the given string is: %s", ar1[i]);
78. }
79.
80.

```

advertisement

\$ cc string22.c

\$ a.out

Enter the string:Welcome to C Programming Class, Welcome Again to C Programming Class!

The longer repeating sequence of the given string is: Programming

Enter the string:Welcome to Sanfoundry, Welcome to C Class

The longer repeating sequence of the given string is: Welcome

185.C Program to Find the Highest Bit Set for any given Integer

[« Prev](#)

[Next »](#)

This is a C Program to find the highest bit set for any given integer.

Problem Description

This C Program finds the Highest Bit Set for any given Integer.

Problem Solution

Take input from the user and finds the highest bit set in a given integer as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to find the Highest Bit Set for any given Integer. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to find the Highest Bit Set for any given Integer
 */
#include <stdio.h>
#define NUM_BITS sizeof(int)*8

int highest_bit_set(int);
void display(int);
int i = NUM_BITS;

void main()
{
    int num, pos;

    printf("\nEnter the number:");
    scanf("%d", &num);

    display(num);
    pos = highest_bit_set(num);
    printf("\nthe position of the highest bit set is %d", pos);
}

/* RETURNS THE POSITION */
int highest_bit_set(int num)
{
    int count = 0;
    while (num >> 1 != 0)
    {
        count++;
        num = num >> 1;
    }
    return(count);
```

```
}

/* DISPLAYS THE NUMBER IN BINARY REPRESENTATION */
void display(int num)
{
    int c;
    c = num & 1;
    if (i > 0)
    {
        i--;
        display(num >> 1);
    }
    printf("%d", c);
}
```

Program Explanation

In this C Program, we are reading the number using the ‘num’ variable. The display() function is used to display the number in binary representation. Binary AND Operator is used to copy a bit to the ‘c’ variable if it exists in both operands.

advertisement

If condition statement is used to check the value of ‘i’ variable value is greater than or equal to 0. If the condition is true then execute the statement and decrement the value of ‘i’ variable by 1. Recursively we call the display() function by moving the value of ‘num’ variable to right by the number of bits specified by the right operand and pass this value as an argument to the display() function.

Once the condition becomes false then print the number in binary representation. The ‘pos’ variable is used to call the highest_bit_set() function by passing the ‘num’ variable value as an argument.

While loop is used to check the condition that the left operand’s value of ‘num’ is moved right by the number of bits specified by the right operand using Binary Right shift operator value is not equal to 0. If the condition is true then execute the loop and increment the value of ‘count’ variable.

advertisement

The ‘num’ variable is used to compute the Binary Right Shift operation by moving the value of ‘num’ to right by the number of bits specified by the right operand and return the value to the called variable ‘pos’. Print the position of the highest bit set value using printf statement.

Runtime Test Cases

```
$ cc bit17.c
$ a.out
enter the number:10000
000000000000000010011100010000
the position of the highest bit set is 13
```

186.C Program to Replace all the Characters by Lowercase

[« Prev](#)

[Next »](#)

This is a C Program to replace all the characters by lowercase.

Problem Description

This is a C Program to replace all the characters by lowercase.

Problem Solution

1. Take a string as input.
2. Use string function strlwr to replace all the characters by lowercase.

advertisement

Program/Source Code

Here is source code of the C Program to replace all the characters by lowercase. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Replace all the Characters by Lowercase
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. int main()
8. {
9.     char string[1000];
10.
11.    printf("Input a string to convert to lower case\n");
12.    gets(string);
13.
14.    printf("Input string in lower case: %s\n",strlwr(string));
15.
16.    return 0;
17.}
```

Program Explanation

1. Take a string as input and store it in the array string[].
2. Using strlwr function replace all the characters by lowercase.

advertisement

Runtime Test Cases

Input a string to convert to lower case
CHANDANA chanikya
rAVELLA

Input string in lower case:
chandana chanikya
ravella

187.C Program to Count the Total Number of Words in the Sentence using Command Line Argument

[« Prev](#)

[Next »](#)

This C Program counts total number of words in the sentence using command line argument.

Here is source code of the C Program to Count the total number of words in the sentence using command line argument. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Count the Total Number of Words in the Sentence
3. * using Command Line Argument
4. */
5. #include <stdio.h>
6.
7. int main(int argc, char *argv[])
8. {
9.     int i = 0;
10.
11.    /* If no sentence is given in the command line */
12.    if (argc == 1)
13.    {
14.        printf("\n No sentence given on command line");
15.        return;
16.    }
17.    else
18.    {
19.        printf("\nThe words in the sentence are:");
20.        /*
21.         * From argv[1] to argv[argc -1] calculate the number of arguments
22.         */
23.        for (i = 1;i < argc;i++)
24.        {
25.            printf("\n% s", argv[i]);
26.        }
27.        printf("\n\nTotal number of words:");
28.        printf(" %d", argc-1);
29.    }
30. }
```

advertisement

```
$ gcc arg1.c
$ a.out Welcome to C Class
```

The words **in** the sentence are:
Welcome
to
C
Class

Total number of words: 4

a.out

188.C Program to Check if a given Integer is Power of 2 using Bitwise Operators

[« Prev](#)

[Next »](#)

This is a C Program to check if a given integer is power of 2 using bitwise operators.

Problem Description

This C Program checks if a given integer is power of 2 using bitwise operators.

Problem Solution

Take input from the user and performs bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to check if a given integer is power of 2 using bitwise operators. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Check if a given Integer is Power of 2 using Bitwise Operators
 */
#include <stdio.h>
#define NUM_BITS_INT (8*sizeof(int))

int power_of_2(unsigned int);

int main()
{
    unsigned int num;

    printf("\nEnter Number");
    scanf("%d", &num);
    power_of_2(num);
}

/*
 * Finding the power of 2 using bit wise operators
 */
int power_of_2(unsigned int x)
{
    int i, count = 0, result, shift_num;

    for (i = 0;i <= NUM_BITS_INT;i++)
    {
        shift_num = x >> i;
        result = shift_num & 1;
        if (result == 1)
            count++;
    }
}
```

```
/*
*If number of bits set to 1 are odd then the number is power of 2
*If number of bits set to 0 are even then the number is not power of 2
*/
if (count % 2 == 1)
    printf("YES");
else
    printf("NO");
}
```

Program Explanation

In this C Program, we are reading the number using ‘num’ variable. The power_of_2() function is used for finding the power of 2 using bit wise operators. Binary Right Shift operator the left operands value is moved right by the number of bits specified by the right operands and assign the value to ‘shift_num’ variable.

advertisement

The ‘result’ variable is used compute the Binary AND operation by copying a bit to the result if it exists in both operands. If condition statement is used to check the value of ‘res’ variable is equal to 1. If the condition is true then execute the statement and increment the value of ‘count’ variable.

If else condition statement is used to check that the number of bits set to 1 is odd and print the number is power of 2. Otherwise, if the condition is false and print the number is not power of 2 and displays the output of the program.

Runtime Test Cases

```
$ gcc bit25.c
$ a.out
Enter Number128
YES
$ a.out
Enter Number126
NO
```

189.C Program to Find the Length of the Longest Repeating Sequence in a String

[« Prev](#)

[Next »](#)

This C Program finds the length of longest repeating sequence in a string.

Here is source code of the C Program to find the Length of the longest repeating sequence in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Length of the Longest Repeating Sequence in a String
3. */
4. #include <stdio.h>
5. #include <string.h>
6. char string[100], words[100][100];
7. int len = 0, word_cnt = 0;
8. int main()
9. {
10.    int i, j = 0, k, mlen = 0, rlen = 0, s = 0, c = 0, m = 0;
11.    printf("\nEnter the string:");
12.    scanf(" %[^\n]s", string);
13.    for (len = 0; string[len] != '\0'; len++);
14.    /*
15.     * Storing the individual words in an array
16.     */
17.    for (k = 0; k < len; k++)
18.    {
19.        if (string[k] != ' ')
20.        {
21.            words[s][j] = string[k];
22.            j++;
23.        }
24.        if (string[k] == ' ')
25.        {
26.            words[s][j] = '\0';
27.            j = 0;
28.            s++;
29.            word_cnt++;
30.        }
31.    }
32.    word_cnt++;
33.    /*
34.     * Compare on Word basis if same word is repeated then check next word & so on
35.     * Increment a counter when consecutive words are repeated
36.     */
37.    for (i = 0; i <= word_cnt; i++)
38.    {
39.        len = 0;
40.        for (j = i+1; j <= word_cnt-1; j++)
41.        {
42.            if (strcmp(words[i], words[j]) != 0)
43.            {
44.                continue;
45.            }
46.            else if (strcmp(words[i], words[j]) == 0)
47.            {
48.                len++;
49.                for (k = i+1, m = j+1; k < j; k++, m++)
50.                {
51.                    if (strcmp(words[k], words[m]) == 0)
52.                    {
53.                        len++;
54.                    }
55.                }
56.            }
57.        }
58.    }
59. }
```

```

54.         continue;
55.     }
56.     else
57.     {
58.         break;
59.     }
60. }
61. if (rlen < len)
62. {
63.     rlen = len;
64.     len = 0;
65. }
66. len = 0;
67. }
68. /*
69. * Finding length of Longest Repeated Sequence
70. */
71. if (mlen < rlen)
72. {
73.     s = i;
74.     mlen = rlen;
75. }
76. }
77. }
78. printf("\nLength of Longest Repeating Sequence:%d\n", mlen);
79. printf("\nTotal number of words : %d", mlen);
80. printf("\n The subsequence is : ");
81. for (i = s, j = 0;j < mlen;i++, j++)
82. printf(" %s", words[i]);
83. char subseq[100];
84. // concat all the words
85. for (i = s, j = 0; j < mlen; i++, j++)
86. strcat (subseq, words[i]);
87. printf ("\nThe length of longest common subsequence is: %d", strlen(subseq)+1);
88. printf("\n");
89. }

```

advertisement

Enter the string:

Welcome to C Programming Class, Welcome Again to C Programming Class!

Length of Longest Repeating Sequence:3

Total number of words : 3

The subsequence is : to C Programming

The length of longest common subsequence is : 19

190.C Program to Implement Regular Expression Matching

[« Prev](#)

[Next »](#)

This C Program implements regular expression matching.

Here is source code of the C Program to implement regular expression matching. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Implements Regular Expression Matching
3. */
4. #include <stdio.h>
5. #include <string.h>
6. #define MATCH printf("\nThe Text Matches The Regular Expression");
7. #define NOTMATCH printf("\nThe Text Doesn't match the Regular Expression");
8.
9. char reg[20], text[20];
10.
11.int main()
12.{
13.    int i, rlen, tlen, f = 0;
14.    char ans;
15.
16.    do {
17.        printf("\nEnter the Regular Expression\n");
18.        scanf(" %[^\n]s", reg);
19.        for (rlen = 0; reg[rlen] != '\0'; rlen++);
20.        printf("\nEnter the text\n");
21.        scanf(" %[^\n]s", text);
22.        for (tlen = 0; text[tlen] != '\0'; tlen++);
23.        if (reg[0] == '*')
24.        {
25.            printf("\nInvalid regular expression");
26.        }
27.        /*
28.         *If the regular expression starts with Alphabet
29.         */
30.        if ((reg[0] >= 65 && reg[0] <= 90) || (reg[0] >= 97 && reg[0] <= 122))
31.        {
32.            if (reg[0] == text[0])
33.            {
34.                switch (reg[1])
35.                {
36.                    case '.':
37.                        switch (reg[2])
38.                        {
39.                            case '*':
40.                                if (tlen != 1)
41.                                {
42.                                    if (reg[3] == text[tlen-1])
43.                                    {
44.                                        MATCH;
45.                                    }
46.                                    else
47.                                    {
48.                                        NOTMATCH;
49.                                    }
50.                                }
51.                            else
52.                            {
53.                                NOTMATCH;
54.                            }
55.                        break;
56.                }
57.            }
58.        }
59.    }
60.}
```

```

56.         case '+':
57.             if (text[1] != reg[3])
58.             {
59.                 if (reg[3] == text[tlen - 1])
60.                 {
61.                     MATCH;
62.                 }
63.                 else
64.                 {
65.                     NOTMATCH;
66.                 }
67.             }
68.             break;
69.         case '?':
70.             if (text[1] == reg[3] || text[2] == reg[3])
71.             {
72.                 if (text[1] == reg[3] || text[2] == reg[3])
73.                 {
74.                     MATCH;
75.                 }
76.                 else
77.                 {
78.                     NOTMATCH;
79.                 }
80.             }
81.             else
82.             {
83.                 NOTMATCH;
84.             }
85.             break;
86.         }
87.         break;
88.     case '*':
89.         if (reg[rlen-1] == text[tlen-1])
90.         {
91.             for (i = 0;i <= tlen-2;i++)
92.             {
93.                 if(text[i] == reg[0])
94.                 {
95.                     f = 1;
96.                 }
97.                 else
98.                 {
99.                     f = 0;
100.                }
101.            }
102.            if (f == 1)
103.            {
104.                MATCH;
105.            }
106.            else
107.            {
108.                NOTMATCH;
109.            }
110.        }
111.        else
112.        {
113.            NOTMATCH;
114.        }
115.        break;
116.    case '+':
117.        if (tlen <= 2)
118.        {
119.            NOTMATCH;
120.        }
121.        else if (reg[rlen-1] == text[tlen-1])

```

```

122.         {
123.             for (i = 0;i < tlen-2;i++)
124.             {
125.                 if (text[i] == reg[0])
126.                 {
127.                     f = 1;
128.                 }
129.                 else
130.                 {
131.                     f = 0;
132.                 }
133.             }
134.
135.             if (f == 1)
136.             {
137.                 MATCH;
138.             }
139.             else
140.             {
141.                 NOTMATCH;
142.             }
143.         }
144.         break;
145.     case '?':
146.         if (reg[rlen -1] == text[tlen-1])
147.         {
148.             MATCH;
149.         }
150.         else
151.         {
152.             NOTMATCH;
153.         }
154.         break;
155.     }
156.
157. }
158. else
159.     printf("Does not match");
160. }
161. /*
162. *If Regular Expression starts with '^'
163. */
164. else if (reg[0] == '^')
165. {
166.     if (reg[1] == text[0])
167.     {
168.         MATCH;
169.     }
170.     else
171.     {
172.         NOTMATCH;
173.     }
174. }
175. /*
176. *If Regular Expression Ends with '$'
177. */
178. else if (reg[rlen-1] == '$')
179. {
180.     if (reg[rlen-2] == text[rlen-1])
181.     {
182.         MATCH;
183.     }
184.     else
185.     {
186.         NOTMATCH;
187.     }

```

```
188. }
189.
190. else
191.     printf("Not Implemented");
192.     printf("\nDo you want to continue?(Y/N)");
193.     scanf(" %c", &ans);
194. } while (ans == 'Y' || ans == 'y');
195. }
```

advertisement

```
$gcc -o regex regular.c
$ ./regex
```

Enter the Regular Expression
C.*g

Enter the text
Cprogramming

The Text Matches The Regular Expression
Do you want to **continue?**(Y/N)y

Enter the Regular Expression
C*g

Enter the text
Cprogramming

The Text Doesn't match the Regular Expression
Do you want to continue?(Y/N)

Enter the Regular Expression
C?.*g

Enter the text
Cprogramming

The Text Matches The Regular Expression
Do you want to continue?(Y/N)y

Enter the Regular Expression
C.?g

Enter the text
Cprogramming

The Text Doesn't match the Regular Expression
Do you want to **continue?**(Y/N)y

Enter the Regular Expression
C.+g

Enter the text
Cprogramming

The Text Matches The Regular Expression
Do you want to **continue?**(Y/N)y

Enter the Regular Expression
C+g

Enter the text
Cprogramming

The Text Doesn't match the Regular Expression
Do you want to continue?(Y/N)y

Enter the Regular Expression

`^C.*`

Enter the text

Cprogramming

The Text Matches The Regular Expression

Do you want to continue?(Y/N)y

Enter the Regular Expression

`^p.*`

Enter the text

Cprogramming

The Text Doesn't match the Regular Expression

Do you want to continue?(Y/N)y

Enter the Regular Expression

`C.*g$`

Enter the text

Cprogramming

The Text Matches The Regular Expression

Do you want to continue?(Y/N)y

Enter the Regular Expression

`C.*n$`

Enter the text

Cprogramming

The Text Doesn't match the Regular Expression

Do you want to continue?(Y/N)n

191.C Program to Convert the Content of File to UpperCase

[« Prev](#)

[Next »](#)

This C Program converts the content of file to UpperCase.

Here is source code of the C Program to convert the content of file to UpperCase. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Convert the Content of File to UpperCase
3.  */
4. #include <stdio.h>
5.
6. int to_upper_file(FILE *);
7.
8. int main(int argc,char *argv[])
9. {
10.    FILE *fp;
11.    int status;
12.
13.    if (argc == 1)
14.    {
15.        printf("Insufficient Arguments:");
16.        printf("No File name is provided at command line");
17.        return;
18.    }
19.    if (argc > 1)
20.    {
21.        fp = fopen(argv[1],"r+");
22.        status = to_upper_file(fp);
23.
24.        /*
25.         *If the status returned is 0 then the coversion of file content was completed successfully
26.         */
27.
28.        if (status == 0)
29.        {
30.            printf("\n The content of \"%s\" file was successfully converted to upper case\n",argv[1]);
31.            return;
32.        }
33.        /*
34.         * If the status returns is -1 then the conversion of file content was not done
35.         */
36.        if (status == -1)
37.        {
38.            printf("\n Failed to convert");
39.            return;
40.        }
41.    }
42.}
43.
44.*/
45. * convert the file content to uppercase
46. */
47.int to_upper_file(FILE *fp)
48.{ 
49.    char ch;
50.
51.    if (fp == NULL)
52.    {
53.        perror("Unable to open file");
```

```
54.     return -1;
55. }
56. else
57. {
58. /*
59. * Read the file content and convert to uppercase
60. */
61. while (ch != EOF)
62. {
63.     ch = fgetc(fp);
64.     if ((ch >= 'a') && (ch <= 'z'))
65.     {
66.         ch = ch - 32;
67.         fseek(fp,-1,SEEK_CUR);
68.         fputc(ch,fp);
69.     }
70. }
71. return 0;
72. }
73. }
```

```
/* Input file : mydata
$ cat mydata
This is Manish
I had worked in Wipro and Cisco
*/
$ gcc file3.c
$ a.out mydata
The content of "mydata" file was successfully converted to upper case
```

```
/* "mydata" after conversion
$ cat mydata
THIS IS MANISH
I HAD WORKED IN WIPRO AND CISCO
*/
```

192.C Program to Insert Character/Word in any Desired Location in a String

[« Prev](#)

[Next »](#)

This C Program inserts character/word in any desired location in a string.

Here is source code of the C Program to insert character/word in any desired location in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Insert Character/Word in any Desired Location
3. * in a String
4. */
5. #include <stdio.h>
6. #include <string.h>
7.
8. void main()
9. {
10.    int i, j, count = 0, pos, flag = 0;
11.    char s1[100], s2[10], s3[100];
12.    char *ptr1, *ptr2, *ptr3;
13.
14.    printf("\nEnter the String:");
15.    scanf(" %[^\n]", s1);
16.    printf("\nEnter the string to be inserted:");
17.    scanf(" %[^\n]", s2);
18.    printf("\nEnter the position you like to insert:");
19.    scanf("%d", &pos);
20.
21.    ptr1 = s1;
22.    ptr3 = s3;
23.    /*COPYING THE GIVEN STRING TO NEW ARRAY AND INSERTING THE STRING IN NEW ARRAY*/
24.    for (i = 0, j = 0; *ptr1 != '\0'; ptr1++, i++, j++, ptr3++)
25.    {
26.        s3[j] = s1[i];
27.        if (*ptr1 == ' ' && flag != 1)
28.            ++count;
29.        if (flag != 1 && count == pos - 1)
30.        {
31.            flag = 1;
32.            for(ptr2 = s2; *ptr2 != '\0'; ptr2++)
33.            {
34.                s3[++j] = *ptr2;
35.                ptr3++;
36.            }
37.            s3[++j] = ' ';
38.            ptr3++;
39.        }
40.    }
41.    s3[j] = '\0';
42.    printf("\nthe string after modification is\n\n %s\n", s3);
43.}
```

advertisement

```
$ cc string10.c
$ a.out
enter the string:Welcome to Sanfoundry's C Programming Class, Welcome Again to C Class!
enter the word to insert:Sanfoundry's
enter the position you like to insert:3
the string after modification is
```

Welcome to Sanfounsr's **Sanfoundry**'s C Programming Class, Welcome Again to C Class!

193.C Program takes Byte as Input and returns all the Bits between given Positions

[« Prev](#)

[Next »](#)

This is a C Program takes byte as input and returns all the bits between given positions.

Problem Description

This C Program takes byte as input and returns all the bits between given positions.

Problem Solution

Take input from the user and returns positions of a and b as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to take byte as input and returns all the bits between given positions. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program takes Byte as Input and returns all the Bits between
 * given Positions
 */
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[])
{
    int a = 0, b = 0, temp = 0, i = 0, countCast = 0;

    char BYTE_HERE[8];
    int FULL_BYT[8];

    printf ("Enter the BYTE: \n");
    gets(BYTE_HERE);

    if (strlen (BYTE_HERE) < 8) {
        printf ("Enter a full 8-bit value.\n");
        return 0;
    }

    printf ("\nEnter the positions a and b : \n");
    scanf ("%d %d", &a, &b);

    // copy from character array to integer array
    for (i = 0; i < 8; i++)
    {
        // convert the character to integer
        FULL_BYT[i] = BYTE_HERE[i] - '0';
    }

    //just print the bits
    for (i = a; i <= b; i++)
    {
        printf ("%d ", FULL_BYT[i]);
    }
}
```

```
    }
    printf("\nBits between positions a and b are:\n");
}

}
```

Program Explanation

1. In this C program, first we define one integer array and one character array. And also take the input from the user.
2. If statement is used to check whether the input is 8 bit in length or not if the input is less than 8 bit inform and abort.
3. For statement is used to copy each bit from character array and convert it to an integer. FULL_BYTE[i] = BYTE_HERE[i] – ‘0’; is used to convert the character to integer
4. Print the bits by using the print statement.

advertisement

Runtime Test Cases

Test case 1 – Invalid Input

Enter the BYTE:

101

Enter a full 8-bit value.

Test case 2 – Here, the entered positions a and b are 0 & 5

advertisement

Enter the BYTE:

10101010

Enter the positions a and b :

0

5

Bits between positions a and b are: 1 0 1 0 1 0

Test case 3 – Here, the entered positions a and b are 0 & 3

Enter the BYTE:

11110000

Enter the positions a and b :

0

3

Bits between positions a and b are: 1 1 1 1

194.C Program to Input 3 Arguments and Operate Appropriately on the Numbers

[« Prev](#)

[Next »](#)

This is a C Program to input 3 arguments and operate appropriately on the numbers.

Problem Description

This program take 3 arguments as input and operate appropriately on the numbers.

Problem Solution

1. Take two numbers and a operator as 3 arguments.
2. Use switch statement to test the operator.
3. According to the operator, do the operation and exit.

advertisement

Program/Source Code

Here is source code of the C Program to input 3 arguments and operate appropriately on the numbers. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Input 3 Arguments and Operate Appropriately on the
3. * Numbers
4. */
5. #include <stdio.h>
6.
7. void main(int argc, char * argv[])
8. {
9.     int a, b, result;
10.    char ch;
11.
12.    printf("arguments entered: \n");
13.    a = atoi(argv[1]);
14.    b = atoi(argv[2]);
15.    ch = *argv[3];
16.    printf("%d %d %c", a, b, ch);
17.    switch (ch)
18.    {
19.        case '+':
20.            result = a + b;
21.            break;
22.        case '-':
23.            result = a - b;
24.            break;
25.        case 'x':
26.            result = a * b;
27.            break;
28.        case '/':
29.            result = a / b;
30.            break;
31.        default:
32.            printf("Enter a valid choice");
33.    }
34.    printf("\nThe result of the operation is %d", result);
35.    printf("\n");
36.}
```

Program Explanation

1. Take two numbers and a operator as input and store it in the variables a, b and ch respectively.
2. Using switch statement, test the operator stored in the variable ch.
3. If it is +, then add a & b and break.
4. If it is -, then subtract a & b and break.
5. If it is *, then multiply a & b and break.
6. If it is /, then divide a & b and break.
7. In the default case print it as “Enter a valid choice”.
8. Store the solution got at steps 3-6 in the variable result.
9. Print the variable result as output and exit.

advertisement

Runtime Test Cases

arguments entered:

5 4 +

The result of the operation is 9

arguments entered:

8 7 -

The result of the operation is 1

arguments entered:

9 6 x

The result of the operation is 54

arguments entered:

100 10 /

The result of the operation is 10

195.C Program to Implement strpbrk() Function

« Prev

Next »

This C Program implements strpbrk() function.

Here is source code of the C Program to implement strpbrk() function. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Implement a strpbrk() Function
3. */
4. #include <stdio.h>
5.
6. char* strpbrk(char *, char *);
7.
8. int main()
9. {
10.    char string1[50], string2[50];
11.    char *pos;
12.
13.    printf("Enter the String:\n");
14.    scanf(" %[^\n]s", string1);
15.    printf("\nEnter the Character Set:\n");
16.    scanf(" %[^\n]s", string2);
17.    pos=strpbrk(string1, string2);
18.    printf("%s", pos);
19.}
20.
21./* Locates First occurrence in string s1 of any character in string s2,
22. * If a character from string s2 is found ,
23. * a pointer to the character in string s1 is returned,
24. * otherwise, a NULL pointer is returned.
25. */
26.char* strpbrk(char *string1, char *string2)
27.{
28.    int i, j, pos, flag = 0;
29.    for (i = 0; string1[i] != '\0'; i++);
30.    pos = i;
31.    for (i = 0; string2[i] != '\0'; i++)
32.    {
33.        for (j = 0; string1[j] != '\0'; j++)
34.        {
35.            if (string2[i] == string1[j])
36.            {
37.                if (j <= pos)
38.                {
39.                    pos = j;
40.                    flag = 1;
41.                }
42.            }
43.        }
44.    }
45.    if (flag == 1)
46.    {
47.        return &string1[pos];
48.    }
49.    else
50.    {
51.        return NULL;
52.    }
53.}
```

advertisement

```
$gcc string34.c
$ a.out
```

Enter the String:
C programming Class

Enter the Character Set:
mp
programming Class

196.C Program to Generate Fibonacci Series of N Numbers using Command-Line Argument

[« Prev](#)

[Next »](#)

This is a C program to generate fibonacci series of n numbers using command-line argument.

Problem Description

This C Program generates fibonacci series of n numbers using command-line argument.

Problem Solution

It displays fibonacci series of n numbers using command-line argument as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to generate fibonacci series of n numbers using command-Llne argument. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Generate Fibonacci Series of N Numbers using
 * Command-Line Argument
 */
#include <stdio.h>

void main(int argc, char * argv[])
{
    int n, last = 0, prev = 1, curr, cnt;
    n = atoi(argv[1]);
    printf("Printing first %d fibonacci nos. -> ", n);
    printf("%d ", last);
    printf("%d ", prev);
    cnt = 2;
    while (cnt<= n-1)
    {
        curr = last + prev;
        last = prev;
        prev = curr;
        cnt++;
        printf("%d ", curr);
    }
    printf("\n");
}
```

Program Explanation

In this C program, we are computing first N Fibonacci numbers using command line arguments. The arguments argc and *argv[] are used. Initially assign the first variable value as 0 and second variable value as 1.

advertisement

The `rec_fibonacci()` function is used to compute the Fibonacci series. If condition statement is used to check the value of ‘num’ variable is equal to 2. If the condition is true then exit the function. Print the statement as the first two numbers are already printed.

If the condition is false then execute the else statement. Compute the value of ‘first’ and ‘second’ variable. Assign to third variable and print the Fibonacci series. Then the value of ‘second’ variable is assigned to the value of ‘first’ variable and the value of ‘third’ variable is assigned to ‘second’ variable and decrement the value of ‘num’ variable.

Runtime Test Cases

```
$ gcc arg5.c
$ a.out 10
Printing first 10 fibonacci nos. -> 0 1 1 2 3 5 8 13 21 34
```

197.C Program to Find the First Occurrence of the any Character of String2 in String1

[« Prev](#)

[Next »](#)

This is a C Program to find the first occurrence of the any character of String2 in string1 & also its position.

Problem Description

This program takes two strings as input and finds the first occurrence of the any character of String2 in string1 & also its position.

Problem Solution

1. Take two strings as input.
2. Compare both the strings using two pointers.
3. Print the character which matches first and its position.

advertisement

Program/Source Code

Here is source code of the C Program to find the first occurrence of the any character of String2 in string1 & also its position. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. /*
3. * C Program to Find the First Occurrence of the any Character of
4. * String2 in string1 & also its Position
5. */
6. #include <stdio.h>
7.
8. void main()
9. {
10. char s1[50], s2[10];
11. int i, flag = 0;
12. char *ptr1, *ptr2;
13.
14. printf("\nEnter the string1:");
15. scanf(" %[^\n]s", s1);
16. printf("\nEnter the string2:");
17. scanf(" %[^\n]s", s2);
18.
19. /*COMPARING THE STRING1 CHARACTER BY CHARACTER WITH ALL CHARACTERS OF STRING1*/
20. for (i = 0, ptr1 = s1; *ptr1 != '\0'; ptr1++)
21. {
22.     i++;
23.     for (ptr2 = s2; *ptr2 != '\0'; ptr2++)
24.     {
25.         if (*ptr1 == *ptr2)
26.         {
27.             flag = 1;
28.             break;
29.         }
30.     }
31.     if (flag == 1)
32.         break;
33. }
34.
35. if (flag == 1)
```

```
36.     printf("\nfirst occurrence of character of string2 in string1 is at position:%d and character is %c", i, *ptr2);
37. else
38.     printf("\none of the characters of string1 match with none of characters of string2");
39. }
```

Program Explanation

1. Take two strings as input and store it in the array's s1[] and s2[].
2. Use pointers ptr1 and ptr2 to point the array's s1[] and s2[] respectively.
3. Compare each character of the array s2[] with the array s1. Use the variable i to find the position.
4. Print the character which matches first and the variable i for the position.

advertisement

Runtime Test Cases

enter the string1:C Programming Class

enter the string2:rnp

first occurrence of character of string2 in string1 is at position:3 and character is p

198.C Program to Swap two Integers without using Temporary Variables and Bitwise Operations

[« Prev](#)

[Next »](#)

This is a C Program to swap two integers without using temporary variables and bitwise operations.

Problem Description

This C Program swap two integers without using temporary variables and bitwise operations.

Problem Solution

Take input from the user and swap two integers without using bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to swap two integers without using temporary variables and bitwise operations. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Swap two Integers without using Temporary Variables
 * and Bitwise Operations
 */
#include <stdio.h>

// Function Prototype
void swap(int *, int *);

void main()
{
    int x, y;
    printf("Enter 2 nos: \n");
    scanf("%d %d", &x, &y);
    printf("\nYou have entered x = %d y = %d \n", x, y);
    swap(&x,&y); // passing the 2 nos to the swap function
}

//function to swap the two numbers
void swap(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
    printf("Swapped . . . \n"); // printing the swapped numbers
    printf("x = %d y = %d\n", *a, *b);
}
```

Program Explanation

In this C Program, we are reading the integer value using ‘x’ and ‘y’ variables respectively. The swap function is used to swap the two numbers.

advertisement

Compute the summation of ‘a’ and ‘b’ variable values and the difference of ‘a’ and ‘b’ variable values. Assign the swapped numbers to the ‘a’ and ‘b’ variables. Print the swap of two integers using printf statement.

Runtime Test Cases

```
$ gcc bit28.c
```

```
$ a.out
```

```
Enter 2 nos:
```

```
4
```

```
7
```

```
You have entered x=4 y=7
```

```
Swapped . . . .
```

```
x=7 y=4
```

199.C Program to Count Number of bits set to 0 in an Integer

[« Prev](#)

[Next »](#)

This is a C Program to count number of bits set to 0 in an integer.

Problem Description

This C Program counts number of bits set to 0 in a integer x.

Problem Solution

Take input from the user and performs bits operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to number of bits set to 0 in a integer x. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Count Number of bits set to 0 in a Integer x
 */
#include <stdio.h>

int main ()
{
    int num = 0, i = 0, n = 0, count = 0, zCount = 0;
    printf ("Enter the number: ");
    scanf ("%d", &num);
    n = num;
    while (n)
    {
        count++;
        n = n >> 1;
    }
    for (i = 0; i < count; i++)
    {
        if (((num >> i) & 1) == 0)
        {
            zCount++;
        }
    }
    printf ("\nNumber of bit's set to zero's are: %d\n", zCount);
    return 0;
}
```

Program Explanation

1. In this C Program, we are reading the integer number using the ‘num’ variable.
2. While loop is used to count the number of bits in the given number.
3. For loop is used to check each bit whether the bit is set or unset. If the bit is unset it increment the count.
4. Here zCount indicates the number of bits set to 0.
5. Print the number of bits are unset in an integer using printf statement.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered number is 1.

```
$ gcc count_bits_unset.c -o bits_unset_count  
$ ./bits_unset_count
```

Enter the number: 1

Number of bit's set to zero's are: 0

Test case 2 – Here, the entered number is 2.

advertisement

```
$ gcc count_bits_unset.c -o bits_unset_count  
$ ./bits_unset_count
```

Enter the number: 2

Number of bit's set to zero's are: 1

Test case 3 – Here, the entered number is 7.

```
$ gcc count_bits_unset.c -o bits_unset_count  
$ ./bits_unset_count
```

Enter the number: 7

Number of bit's set to zero's are: 0

Test case 4 – Here, the entered number is 8.

advertisement

```
$ gcc count_bits_unset.c -o bits_unset_count  
$ ./bits_unset_count
```

Enter the number: 8

Number of bit's set to zero's are: 3

200.C Program to Replace First Letter of every Word with Capital Letter

[« Prev](#)

[Next »](#)

This C Program replaces first letter of every word with caps.

Here is source code of the C Program to replace first letter of every word with caps. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to replace first letter of every word with caps
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. void main(int argc, char *argv[])
8. {
9.     FILE *fp1;
10.    int return_val;
11.
12.    if ((fp1 = fopen(argv[1], "r+")) == NULL)
13.    {
14.        printf("file cant be opened");
15.        exit(0);
16.    }
17.    return_val = init_cap_file(fp1);
18.    if (return_val == 1)
19.    {
20.        printf("\nsuccess");
21.    }
22.    else
23.    {
24.        printf("\n failure");
25.    }
26.}
27.
28.int init_cap_file(FILE *fp1)
29.
30.    char ch;
31.
32.    ch = fgetc(fp1);
33.    if (ch >= 97 && ch <= 122)
34.    {
35.        fseek(fp1, -1L, 1);
36.        fputc(ch - 32, fp1);
37.    }
38.    while (ch != EOF)
39.    {
40.        if (ch == ' ' || ch == '\n')
41.        {
42.            ch = fgetc(fp1);
43.            if (ch >= 97 && ch <= 122)
44.            {
45.                fseek(fp1, -1L, 1);
46.                fputc(ch - 32, fp1);
47.            }
48.        }
49.        else
50.            ch = fgetc(fp1);
51.    }
52.    return 1;
53.}
```

```
$ vi file5test  
$ cat file5test  
chandana ravella  
chanikya ravella  
sree lakshmi ravella  
sree ramulu ravella  
$ cc file5.c  
$ a.out file5test
```

```
success$ cat file5test  
Chandana Ravella  
Chanikya Ravella  
Sree Lakshmi Ravella  
Sree Ramulu Ravella
```

201.C Program to Compare two Binary Files, Printing the First Byte Position where they Differ

[« Prev](#)

[Next »](#)

This C Program compares two binary files, printing the first byte position where they differ.

Here is source code of the C Program to compare two binary files, printing the first byte position where they differ. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Compare two Binary Files, Printing the First Byte
3.  * Position where they Differ
4. */
5. #include <stdio.h>
6.
7. void compare_two_binary_files(FILE *,FILE *);
8.
9. int main(int argc, char *argv[])
10. {
11.     FILE *fp1, *fp2;
12.
13.     if (argc < 3)
14.     {
15.         printf("\nInsufficient Arguments: \n");
16.         printf("\nHelp:./executable <filename1> <filename2>\n");
17.         return;
18.     }
19.     else
20.     {
21.         fp1 = fopen(argv[1], "r");
22.         if (fp1 == NULL)
23.         {
24.             printf("\nError in opening file %s", argv[1]);
25.             return;
26.         }
27.
28.         fp2 = fopen(argv[2], "r");
29.
30.         if (fp2 == NULL)
31.         {
32.             printf("\nError in opening file %s", argv[2]);
33.             return;
34.         }
35.
36.         if ((fp1 != NULL) && (fp2 != NULL))
37.         {
38.             compare_two_binary_files(fp1, fp2);
39.         }
40.     }
41. }
42.
43.*/
44. /* compare two binary files character by character
45. */
46. void compare_two_binary_files(FILE *fp1, FILE *fp2)
47. {
48.     char ch1, ch2;
49.     int flag = 0;
50.
51.     while (((ch1 = fgetc(fp1)) != EOF) && ((ch2 = fgetc(fp2)) != EOF))
52.     {
53.         /*
```

```

54.     * character by character comparision
55.     * if equal then continue by comparing till the end of files
56.     */
57.     if(ch1 == ch2)
58.     {
59.         flag = 1;
60.         continue;
61.     }
62.     /*
63.     * If not equal then returns the byte position
64.     */
65.     else
66.     {
67.         fseek(fp1, -1, SEEK_CUR);
68.         flag = 0;
69.         break;
70.     }
71. }
72.
73. if(flag == 0)
74. {
75.     printf("Two files are not equal : byte poistion at which two files differ is %d\n", ftell(fp1)+1);
76. }
77. else
78. {
79.     printf("Two files are Equal\n ", ftell(fp1)+1);
80. }
81.}

```

advertisement

```

$ gcc file15.c
$ a.out /bin/chgrp /bin/chown
Two files are not equal : byte poistion at which two files differ is 25

```

```

/*
 * Verify using cmp command
*/
$ cmp /bin/chgrp /bin/chown
/bin/chgrp /bin/chown differ: byte 25, line 1

$ a.out a.out a.out
Two files are Equal
/*
 * Verify using cmp command
*/
$ cmp a.out a.out

```

202.C Program to Replace Bits in Integer from Specified Positions from Another Integer

[« Prev](#)

[Next »](#)

This is a C Program to replace bits in integer from specified positions from another integer

Problem Description

This C Program counts total number of words in the sentence using command line argument.

Problem Solution

Take input from the user and counts total number of words in the sentence as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Count the total number of words in the sentence using command line argument. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Replace Bits in Integer from Specified Positions from
 * Another Integer
 */
#include <stdio.h>

int main ()
{
    int num1 = 0, num2 = 0, i = 0, j = 0, xor = 0, res = 0;
    printf ("Enter the first number: ");
    scanf ("%d", &num1);
    printf ("\nEnter the second number: ");
    scanf ("%d", &num2);
    printf ("Enter the i'th bit in num1 to replace with j'th bit in num2: ");
    scanf ("%d", &i);
    printf ("\nEnter the j'th bit in num2 to replace with i'th bit in num1: ");
    scanf ("%d", &j);
    printf ("\n");
    if (num1 == num2 && i == j)
    {
        printf ("%d", num1);
        printf ("\n");
        return 0;
    }
    xor = ((num1 >> i) ^ (num2 >> j)) & 1;
    res = num1 ^ (xor << i) ^ (xor ^ j);
    printf ("\nResult = %d\n", res);
    return 0;
}
```

Program Explanation

1. Take the input numbers from the user and store it in num1 & num2 variables. Prompt user to enter the position of ith bit in num1 which they want to replace jth bit in num2.
2. If loop is used to check whether both the numbers entered by the user are the same and also i == j. This gives us no effect, hence it return.

3. Fetch the jth bit from the num2 and ith bit from num1 by using $xor = ((num1 >> i) \wedge (num2 >> j)) \& 1;$
4. Now replace the ith bit in num1 with that of jth bit in num2 by using $res = num1 \wedge (xor \ll i) \wedge (xor \wedge j);$
5. Print the result from res variable using print statement.

advertisement

Runtime Test Cases

Test case 1 – Here, the first and second numbers are same.

```
$ gcc replace_bits.c -o replace  
$ ./replace
```

Enter the first number: 10

Enter the second number: 10

Enter the i'th bit in num1 to replace with j'th bit in num2: 2

Enter the j'th bit in num2 to replace with i'th bit in num1: 1

Result = 14

Test case 2 – Here, the first and second numbers are single digit numbers.

advertisement

```
$ gcc replace_bits.c -o replace  
$ ./replace
```

Enter the first number: 2

Enter the second number: 9

Enter the i'th bit in num1 to replace with j'th bit in num2: 1

Enter the j'th bit in num2 to replace with i'th bit in num1: 2

Result = 3

Test case 3 – Here, the first and second numbers are double digit numbers.

```
$ gcc replace_bits.c -o replace  
$ ./replace
```

Enter the first number: 99

Enter the second number: 100

Enter the i'th bit in num1 to replace with j'th bit in num2: 2

Enter the j'th bit in num2 to replace with i'th bit in num1: 5

Result = 99

203.C Program to Swap the ith and jth Bits for a 32-Bit Integer

[« Prev](#)

[Next »](#)

This is a C Program to swap the ith and jth bits for a 32-bit integer.

Problem Description

This C Program swaps the ith and jth bits for a 32-bit integer.

Problem Solution

advertisement

Take input from the user and swaps the ith and jth bits as shown in the program below.

Program/Source Code

Here is source code of the C Program to swap the ith and jth bits for a 32-bit integer. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Swap the ith and jth Bits for a 32-Bit Integer
 */
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int n = 0, num = 0, p = 0, q = 0;

    printf ("Enter the number: ");
    scanf ("%d", &n);

    printf ("\nEnter position 1: ");
    scanf ("%d", &p);

    printf ("\nEnter position 2: ");
    scanf ("%d", &q);

    num = n;

    // see if the bits are same. we use XOR operator to do so.
    if (((n & (1 << p)) >> p) ^ ((n & (1 << q)) >> q))
    {
        n ^= (1 << p);
        n ^= (1 << q);
    }

    printf ("\nThe result after swapping the respective bits are: %d\n", n);

    return 0;
}
```

Program Explanation

1. In this C Program, we are reading the number using ‘n’ variable and also the bit positions using ‘p’ and ‘q’ variables respectively.

2. First shift the bit in given position to right-end. This can be achieved by the code below.

for p'th bit – $n \& (1 << p) >> p$

for q'th bit – $(n \& (1 << q)) >> q$

3. Next step is to perform XOR operation. If the bits are the same, no need to swap.

4. If the bits are not the same, just toggle the bits. This can be achieved by the code below.

toggle bit at position p: $n ^= (1 << p)$; toggle bit at position q: $n ^= (1 << q)$;

Runtime Test Cases

advertisement

Test case 1 – Here, the entered number is 1. Postion 1 is 0 & Postion 2 is 1.

```
./a.out  
Enter the number: 2
```

Enter position 1: 0

Enter position 2: 1

The result after swapping the respective bits are: 1

Test case 2 – Here, the entered number is 8. Postion 1 is 0 & Postion 2 is 3.

advertisement

```
./a.out  
Enter the number: 8
```

Enter position 1: 0

Enter position 2: 3

The result after swapping the respective bits are: 1

Test case 3 – Here, the entered number is 11. Postion 1 is 0 & Postion 2 is 4.

advertisement

```
./a.out  
Enter the number: 11
```

Enter position 1: 0

Enter position 2: 4

The result after swapping the respective bits are: 26

204.C Program to Count No of Lines, Blank Lines, Comments in a given Program

[« Prev](#)

[Next »](#)

This is a C Program to Count No of Lines, Blank Lines, Comments in a given Program.

Problem Description

This C Program counts the no of lines, blank lines, comments in a given program.

Problem Solution

1. First count the number of lines in a file.
2. Count the number of blank lines.
3. Use the while loop for step 1-2.
4. Use another while loop to count the number of comment lines in a file.
5. Use fseek function to alter the position in the file.

advertisement

Program/Source Code

Here is source code of the C Program to count no of lines, blank lines, comments in a given program. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Count No of Lines, Blank Lines, Comments in a given Program
3. */
4. #include <stdio.h>
5.
6. void main(int argc, char* argv[])
7. {
8.     int line_count = 0, n_o_c_1 = 0, n_o_n_b_1 = 0, n_o_b_1 = 0, n_e_c = 0;
9.     FILE *fp1;
10.    char ch;
11.    fp1 = fopen(argv[1], "r");
12.
13.    while ((ch = fgetc(fp1)) != EOF)
14.    {
15.        if (ch == '\n')
16.        {
17.            line_count++;
18.        }
19.        if (ch == '\n')
20.        {
21.            if ((ch = fgetc(fp1)) == '\n')
22.            {
23.                fseek(fp1, -1, 1);
24.                n_o_b_1++;
25.            }
26.        }
27.        if (ch == ';')
28.        {
29.            if ((ch = fgetc(fp1)) == '\n')
30.            {
31.                fseek(fp1, -1, 1);
32.                n_e_c++;
33.            }
34.        }
35.    }
36. }
```

```

33.     }
34.   }
35. }
36. fseek(fp1, 0, 0);
37. while ((ch = fgetc(fp1))!= EOF)
38. {
39.   if(ch == ';')
40.   {
41.     if((ch = fgetc(fp1)) == ';')
42.     {
43.       n_o_c_l++;
44.     }
45.   }
46. }
47. printf("Total no of lines: %d\n", line_count);
48. printf("Total no of comment line: %d\n", n_o_c_l);
49. printf("Total no of blank lines: %d\n", n_o_b_l);
50. printf("Total no of non blank lines: %d\n", line_count-n_o_b_l);
51. printf("Total no of lines end with semicolon: %d\n", n_e_c);
52.

```

Program Explanation

1. Open the file and point it to the file pointer fp1.
2. Initialize the variables line_count, n_o_c_l, n_o_n_b_l, n_o_b_l, n_e_c to zero.
3. Using while loop read the next line character and store it in the variable ch. Do this until EOF.
4. Inside the loop and using if,else statements count the number of lines in the file and store it in the variable line_count.
5. Count of number of blank lines and store it in the variable n_o_b_l.
6. Check if the variable ch is equal to ;. If it is, then increment the variable n_e_c.
7. Use another while loop to count the number of comment lines and store it the variable n_o_c_l.
8. For the number of non blank lines subtract line_count from n_o_b_l.
9. Print the variables and exit.

advertisement

Runtime Test Cases

```

Total no of lines: 204
Total no of comment line: 19
Total no of blank lines: 11
Total no of non blank lines: 193
Total no of lines end with semicolon: 66

```

205.C Program to Replace Bits in Integer x from Bit Position a to b from another Integer y

[« Prev](#)

[Next »](#)

This is a C Program to replace bits in integer x from bit position a to b from another integer y.

Problem Description

This C Program replace bits in integer x from bit position a to b from another integer y.

Problem Solution

Take input from the user and performs bits operation as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to replace bits in integer x from bit position a to b from another integer y. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Replace Bits in Integer x from Bit Position a to b from another Integer y
 */
#include <stdio.h>

void changebits(int, int, int, int);

int main()
{
    int num1, num2, pos1, pos2;

    printf("Replacing the bits in integer x from bit position a to b from another integer y\n");
    printf("read number 1\n");
    scanf("%x", &num1);
    printf("Read number 2:\n");
    scanf("%x", &num2);
    printf("Read LSB position:\n");
    scanf("%d", &pos1);
    printf("MSB should always be greater than LSB\n");
    printf("Read MSB position:\n");
    scanf("%d", &pos2);
    changebits(num1, num2, pos1, pos2);
    return 0;
}

/*Function to swap bits in given positions*/

void changebits(int num1, int num2, int pos1, int pos2)
{
    int temp1, temp_1, buffer2, bit1 = 0, bit2 = 0, counter = 0, a = 1;

    temp1 = num1;
    temp_1 = num1;
    buffer2 = num2;
    for (pos1 <= pos2; pos1++)
    {
        if (a == 1)
        {
            if (bit1 == 1)
                temp1 |= (1 << pos1);
            else
                temp1 = temp1 & ~(1 << pos1);
        }
        else
        {
            if (bit2 == 1)
                temp1 |= (1 << pos1);
            else
                temp1 = temp1 & ~(1 << pos1);
        }
        a++;
    }
    num1 = temp1;
    num2 = buffer2;
}
```

```

{
    a = 1;
    num1 = temp_1;
    num2 = buffer2;
    while (counter <= pos1)
    {
        if (counter == pos1)
            bit1 = (num1&1); //placing the bit of position 1 in bit1
        counter++;
        num1>> = 1;
    }
    counter = 0;
    while (counter <= pos1)
    {
        if (counter == pos1)
            bit2 = (num2&1); //placing the bit of position 2 in bit2
        counter++;
        num2 >>= 1;
    }
    counter = 0;
    if (bit1 == bit2);
    else
    {
        while (counter++<pos1)
            a = a << 1;
        temp1 ^= a; //placing the replaced bit integer into temp1 variable
    }
    counter = 0;
}
printf("the number num1 after shifting the bits is 0x%x\n", temp1);
}

```

Program Explanation

This C program, we are reading two integer numbers using ‘num1’ and ‘num2’ variables respectively, and the Least Significant Bit (LSB) and Most Significant Bit (MSB) using ‘pos1’ and ‘pos2’ variables respectively. The changebits() function is used to swap bits in given positions. Interchange the value of ‘num1’ variable to ‘temp1’ variable and the value of ‘num2’ variable to ‘buffer2’ variable.

advertisement

For loop is used for replacing the bits, assign the value of ‘temp1’ variable to ‘num1’ and the value of ‘buffer2’ variable to ‘num2’. While is used for placing the bit of position 1 in bit1. Check the value of ‘counter’ variable is less than or equal to the value of ‘pos1’ variable.

If the condition is true then execute the while loop. If condition statement is used to check that the value of ‘counter’ variable is equal to the value of ‘pos1’ variable. Place the bit of position 1 in bit1 and increment the value of ‘counter’ variable. Using binary right shift operator the 1 value is moved right by the number of bits specified by the value of ‘num1’ variable and assign to ‘num1’ variable.

Another while loop is used for placing the bit of position 2 in bit2, Using binary right shift operator the 1 value is moved right by the number of bits specified by the value of ‘num2’ variable and assign to ‘num2’ variable. Then if-else condition statement is used for placing the replaced bit integer ‘a’ into ‘temp1’ variable. Print the replaced bits in integer x from bit position ‘a’ to ‘b’ from another Integer y.

advertisement

Runtime Test Cases

```
$ cc bit13.c
$ a.out
```

Replacing the bits in integer x from bit position a to b from another integer y

read number 1

0x11223344

Read number 2:

0x55667788

Read LSB position:

12

MSB should always be greater than LSB

Read MSB position:

19

the number num1 after shifting the bits is 0x11267344

206.C Program to Reverse the Contents of a File and Print it

[« Prev](#)

[Next »](#)

This C Program reverses the contents of a file and print it.

Here is source code of the C Program to reverse the contents of a file and print it. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Reverse the Contents of a File and Print it
3. */
4. #include <stdio.h>
5. #include <errno.h>
6.
7. long count_characters(FILE *);
8.
9. void main(int argc, char * argv[])
10. {
11.     int i;
12.     long cnt;
13.     char ch, ch1;
14.     FILE *fp1, *fp2;
15.
16.     if (fp1 = fopen(argv[1], "r"))
17.     {
18.         printf("The FILE has been opened...\n");
19.         fp2 = fopen(argv[2], "w");
20.         cnt = count_characters(fp1); // to count the total number of characters inside the source file
21.         fseek(fp1, -1L, 2); // makes the pointer fp1 to point at the last character of the file
22.         printf("Number of characters to be copied %d\n", ftell(fp1));
23.
24.         while (cnt)
25.         {
26.             ch = fgetc(fp1);
27.             fputc(ch, fp2);
28.             fseek(fp1, -2L, 1); // shifts the pointer to the previous character
29.             cnt--;
30.         }
31.         printf("\n**File copied successfully in reverse order**\n");
32.     }
33.     else
34.     {
35.         perror("Error occurred\n");
36.     }
37.     fclose(fp1);
38.     fclose(fp2);
39. }
40.// count the total number of characters in the file that *f points to
41.long count_characters(FILE *f)
42.{
43.    fseek(f, -1L, 2);
44.    long last_pos = ftell(f); // returns the position of the last element of the file
45.    last_pos++;
46.    return last_pos;
47.}
```

advertisement

```
$ gcc file12.c
```

```
$ cat test2
```

The **function** STRERROR returns a pointer to an ERROR MSG STRING whose contents are implementation defined.
THE STRING is not MODIFIABLE and maybe overwritten by a SUBSEQUENT Call to the STRERROR function.

```
$ a.out test2 test_new
```

The FILE has been opened..

```
Number of characters to be copied 203
```

```
**File copied successfully in reverse order**
```

```
$ cat test_new
```

```
.noitcnuf RORRERTS eht ot llaC TNEUQESBUS a yb nettirvrevo ebyam dna ELBAIFIDOM ton si GNIRTS EHT  
.denified noitatnemelpmi era stnetnoc esohw GNIRTS GSM RORRE na ot retniop a snruter RORRERTS noitcnuf ehT
```

```
$ ./a.out test_new test_new_2
```

```
The FILE has been opened..
```

```
Number of characters to be copied 203
```

```
**File copied successfully in reverse order**
```

```
$ cat test_new_2
```

```
The function STRERROR returns a pointer to an ERROR MSG STRING whose contents are implementation defined.
```

```
THE STRING is not MODIFIABLE and maybe overwritten by a SUBSEQUENT Call to the STRERROR function.
```

```
$ cmp test test_new_2
```

207.C Program to Convert the Content of File to LowerCase

[« Prev](#)

[Next »](#)

This C Program converts the content of file to lowercase.

Here is source code of the C Program to convert the content of file to lowercase. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Convert the Content of File to LowerCase
3. */
4. #include <stdio.h>
5. #include <errno.h>
6.
7. int to_lower_file(FILE *);
8.
9. void main(int argc, char * argv[])
10. {
11.     int op = -1;
12.     char ch;
13.     FILE *fp;
14.     if (fp = fopen(argv[1], "r+"))
15.     {
16.         printf("FILE has been opened..!!!\n");
17.         op = to_lower_file(fp);
18.         printf(" %d \n", op);
19.         fclose(fp);
20.     }
21.     else
22.     {
23.         perror("Error Occured");
24.         printf(" %d\n ", op);
25.     }
26. }
27.
28.int to_lower_file(FILE *f)
29.{
30.    int c;
31.    char ch;
32.    while ((ch = fgetc(f))!= EOF)
33.    {
34.        c = (int)ch;
35.        if (c >= 65 && c <= 90)
36.        {
37.            ch = ch + 32;
38.            fseek(f, -1L, 1);
39.            fputc(ch, f);
40.        }
41.    }
42.    return 0;
43.}

```

advertisement

```

$ gcc file4.c
$ cat test1
THE FUNCTION STRERROR RETURNS A POINTER TO AN ERROR MSG STRING WHOSE CONTENTS ARE
IMPLEMENTATION DEFINED.
THE STRING IS NOT MODIFIABLE AND MAYBE OVERWRITTEN BY A SUBSEQUENT CALL TO THE STRERROR
FUNCTION.
$ ./a.out test1
FILE has been opened..!!!
0
$ cat test1
the function strerror returns a pointer to an error msg string whose contents are implementation defined.
the string is not modifiable and maybe overwritten by a subsequent call to the strerror function

```

208.C Program to find the possible subsets of the String

[« Prev](#)

[Next »](#)

This C Program finds possible subsets of the String.

Here is source code of the C Program to find the possible subsets of the String. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to find the possible subsets of the String
```

```

3. */
4. #include <stdio.h>
5.
6. char string[50], n;
7. void subset(int, int, int);
8.
9. int main()
10. {
11.     int i, len;
12.
13.     printf("Enter the len of main set : ");
14.     scanf("%d", &len);
15.     printf("Enter the elements of main set : ");
16.     scanf("%s", string);
17.     n = len;
18.     printf("The subsets are :\n");
19.     for (i = 1;i <= n;i++)
20.         subset(0, 0, i);
21. }
22.
23./*Function to find the number of subsets in the given string*/
24.
25.void subset(int start, int index, int num_sub)
26.{
27.    int i, j;
28.    if (index - start + 1 == num_sub)
29.    {
30.        if (num_sub == 1)
31.        {
32.            for (i = 0;i < n;i++)
33.                printf("%c\n", string[i]);
34.        }
35.        else
36.        {
37.            for (j = index;j < n;j++)
38.            {
39.                for (i = start;i < index;i++)
40.                    printf("%c", string[i]);
41.                printf("%c\n", string[j]);
42.            }
43.            if (start != n - num_sub)
44.                subset(start + 1, start + 1, num_sub);
45.        }
46.    }
47.    else
48.    {
49.        subset(start, index + 1, num_sub);
50.    }
51.}

```

advertisement

```
$ cc string19.c
```

```
$ a.out
```

```
Enter the len of main set : 11
```

```
Enter the elements of main set : programming
```

```
The subsets are :
```

```
p
r
o
g
r
a
m
m
i
n
```

g

pr

po

pg

pr

pa

pm

pm

pi

pn

pg

ro

rg

rr

ra

rm

rm

ri

rn

rg

og

or

oa

om

om

oi

on

og

gr

ga

gm

gm

gi

gn

gg

ra

rm

rm

ri

rn

rg

am

am

ai

an

ag

mm

mi

mn

mg

mi

mn

mg

in

ig

ng

pro

prg

prr

pra

prm

prm

pri

prn

prg

rog

ror
roa
rom
rom
roi
ron
rog
ogr
oga
ogm
ogm
ogi
ogn
ogg
gra
grm
grm
gri
grn
grg
ram
ram
rai
ran
rag
amm
ami
amn
amg
mmi
mmn
mmg
min
mig
ing
prog
pror
proa
prom
prom
proi
pron
prog
rogr
roga
rogm
rogm
rogi
rogn
rogg
ogra
ogrm
ogrm
ogri
ogrn
ogrg
gram
gram
grai
gran
grag
ramm
rami
ramn
ramg
ammi

ammn
ammg
mmin
mmig
ming
progr
proga
progm
progm
progi
progn
progg
rogra
rogrm
rogrm
rogri
rogrn
rogrg
ogram
ogram
ograi
ogran
ograg
gramm
grami
gramn
gramg
rammi
rammn
rammg
ammin
ammig
mming
progra
program
program
progr
progn
progrg
rogram
rogram
rograi
rogran
rorag
ogramm
ogrami
ogramn
ogramg
grammi
grammn
grammg
rammin
rammig
amming
program
program
prograi
progan
prograg
rogramm
rogrami
rogramn
rogramg
ogrammi
ogrammn
ogrammg

```
grammin  
grammig  
ramming  
programm  
programi  
programn  
programg  
rogrammi  
rogrammn  
rogrammg  
ogrammin  
ogrammig  
gramming  
programmi  
programmn  
programmg  
rogrammin  
rogrammig  
ogramming  
programmin  
programmig  
rogramming  
programming
```

209.C Program to Count the Number of Unique Words

[« Prev](#)

[Next »](#)

This C Program Counts the Number of Unique Words.

Here is source code of the C Program to Count the Number of Unique Words. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Count the Number of Unique Words
3. */
4. #include <stdio.h>
5. #include <string.h>
6. #include <stdlib.h>
7. int main()
8. {
9.     int i = 0, e, j, d, k, space = 0;
10.    char a[50], b[15][20], c[15][20];
11.
12.    printf("Read a string:\n");
13.    fflush(stdin);
14.    scanf("%[^%s]", a);
15.    for (i = 0; a[i] != '\0'; i++) //loop to count no of words
16.    {
17.        if (a[i] == ' ')
18.            space++;
19.    }
20.    i = 0;
21.    for (j = 0; j < (space + 1); i++, j++) //loop to store each word into an 2D array
22.    {
23.        k = 0;
24.        while (a[i] != '\0')
25.        {
26.            if (a[i] == ' ')
27.            {
28.                break;
29.            }
30.            else
31.            {
32.                b[j][k++] = a[i];
33.                i++;
34.            }
35.        }
36.        b[j][k] = '\0';
37.    }
38.    i = 0;
39.    strcpy(c[i], b[i]);
40.    for (e = 1; e <= j; e++) //loop to check whether the string is already present in the 2D array or not
41.    {
42.        for (d = 0; d <= i; d++)
43.        {
44.            if (strcmp(c[i], b[e]) == 0)
45.                break;
46.            else
47.            {
48.                i++;
49.                strcpy(c[i], b[e]);
50.                break;
51.            }
52.        }
53.    }
54.    printf("\nNumber of unique words in %s are: %d", a, i);
55.    return 0;
56.}

```

advertisement

```

$ cc string7.c
$ a.out
Read a string:
Welcome to Sanfoundry's C-programming class, Welcome again to C class!
The length of input string is:70

```

Number of unique words in Welcome to Sanfoundry's C-programming class, Welcome again to C cla

210.C Program to Find the Position of String of 1-bits in a Number for a given Length

[« Prev](#)

[Next »](#)

This is a C Program to find the position of string of 1-bits in a number for a given length.

Problem Description

This C Program finds the position of string of 1-bits in a number for a given length.

Problem Solution

Take input from the user and finds string position as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to find the position of string of 1-bits in a number for a given length. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Find the Position of String of 1-bits in a Number
 * for a given Length
 */
#include <stdio.h>

void main()
{
    int n, len, pos = 0, i = 0, count = 0;

    printf("/**Finding the position of 1-bits in a number for given length**\n");
    printf("enter a number\n");
    scanf("%d", &n);
    printf("enter the length\n");
    scanf("%d", &len);
    while (i <= 32)
    {
        if ((n & 1) == 1) //checking whether there is a 1-bit in the current position
        {
            count++; //counting the consecutive 1's in the integer
            pos = i;
            if (count == len) //checking whether the length matches
            {
                break;
            }
        }
        if ((n & 1) == 0)
        {
            count = 0;
        }
        n = n>>1;
        i++;
    }
    printf("the position of 1 in the string : %d\n", pos);
}
```

Program Explanation

In this C Program, we are reading the number and length using ‘n’ and ‘len’ variables respectively. Using while loop assign the position of strings of 1-bits in a number for a given length. If condition statement is used to check that there is a 1-bit in the current position. If the condition is true, then it will execute the statement for counting the consecutive 1’s in the integer.

advertisement

Another if condition statement is used for checking whether the length matches. Another if condition statement is used to check whether there is a 1-bit in the starting position. If the condition is true then execute the statement.

Using Binary Right shift operator, the left operand's value is moved right by the number of bits specified by the right operands and assign the value to n variable then count the consecutive 1's in the integer. Print the position of string of 1-bits in a number for a given length.

Runtime Test Cases

```
$ cc bit7.c
$ a.out
**Finding the position of 1-bits in a number for given length**
enter a number
10000
enter the length
3
the position of 1 in the string : 10
$ a.out
enter a number
700
enter the length
4
the position of 1 in the string : 5
```

211.C Program to Display the Characters in Prime Position a given String

[« Prev](#)

[Next »](#)

This is a C Program to display the characters in prime position of a given string.

Problem Description

This program prints the characters in prime position of a given string.

Problem Solution

1. Take a string as input.
2. Find the number which gets divided only once and consecutively print the character of the obtained position.

advertisement

Program/Source Code

Here is source code of the C Program to display the characters in prime position a given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Display the Characters in Prime Position a given String
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. void main()
8. {
9.     int i, j, k, count = 0;
10.    char str[50];
11.
12.    printf("enter string\n");
13.    scanf("%[^\\n]s", str);
14.    k = strlen(str);
15.    printf("prime characters in a string are\n");
16.    for (i = 2; i <= k; i++)
17.    {
18.        count = 0;
19.        for (j = 2; j <= k; j++)
20.        {
21.            if (i % j == 0)
22.            {
23.                count++;
24.            }
25.        }
26.        if (count == 1)
27.        {
28.            printf("%c\n", str[i - 1]);
29.        }
30.    }
31. }
```

Program Explanation

1. Take a string as input and store it in the array str[].
2. Store the length of the input string in the variable k.
3. Use two loops to divide the numbers upto the value of k.
4. Increment the variable count when the remainder is zero.
5. If the variable count is equal to 1, then print the corresponding character.

advertisement

Runtime Test Cases

```
enter string
welcome to sanfoundry c-programming class!
prime characters in a string are
e
l
o
e
```

```
a  
u  
d  
c  
r  
m  
c  
s
```

212.C Program to find First and Last Occurrence of given Character in a String

[« Prev](#)

[Next »](#)

This is a C Program to find first and last occurrence of given character in a string.

Problem Description

This program takes a string and a character as input and finds the first and last occurrence of the input character in a string.

Problem Solution

1. Take a string and a character as input.
2. Using for loop search for the input character.
3. When the character is found, then print its corresponding position.
4. Again keep on searching for the input character. Now keep on incrementing a variable whenever the input character encounters.
5. Do step-4 until the end of string. when it is done, print the value of the variable.

advertisement

Program/Source Code

Here is source code of the C Program to find first and last occurrence of given character in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to find First and Last Occurrence of given
3. * Character in a String
4. */
5. #include <stdio.h>
6. #include <string.h>
7.
8. void main()
9. {
10.     int i, count = 0, pos1, pos2;
11.     char str[50], key, a[10];
12.
13.     printf("enter the string\n");
14.     scanf(" %[^\n]", str);
15.     printf("enter character to be searched\n");
16.     scanf(" %c", &key);
17.     for (i = 0; i <= strlen(str); i++)
18.     {
19.         if (key == str[i])
20.         {
21.             count++;
22.             if (count == 1)
23.             {
24.                 pos1 = i;
25.                 pos2 = i;
26.                 printf("%d\n", pos1 + 1);
27.             }
28.             else
29.             {
30.                 pos2 = i;
31.             }
32.         }
33.     }
34.     printf("%d\n", pos2 + 1);
35. }
```

Program Explanation

1. Take a string and a character as input and store it in the array str[] and variable key respectively.
2. Using for loop search for the variable key. If it is found then increment the variable count.
3. If the value of count is equal to 1, then copy the value of i into the variables pos1 and pos2 and print the value (pos+1) as the first position.
4. If the value of count is not equal to 1, then just copy the value of i into the variable pos2. Do this step until the end of string.
5. Print the value (pos2+1) as the last position and exit.

Runtime Test Cases

```
enter the string
welcome to sanfoundry's c programming class!
enter character to be searched
m
6
34
```

213.C Program to Find All Possible Subsets of given Length in String

[« Prev](#)

[Next »](#)

This C Program Find all possible subsets of given length in string.

Here is source code of the C Program to Find all possible subsets of given length in string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find All Possible Subsets of given Length in String
3. */
4. #include <stdio.h>
5. #include <string.h>
```

```

6.
7. char a[20];
8. int n, len, j;
9.
10. void main()
11. {
12.     int i, index = 0, start = 0;
13.
14.     printf("Enter the string\n");
15.     scanf("%s", a);
16.     n = strlen(a);
17.     printf("enter input length\n");
18.     scanf("%d", &len);
19.     printf("The subsets are\n");
20.     for (i = 1; i <= n; i++)
21.     {
22.         if (index - start + 1 == i)
23.         {
24.             if (i == len)
25.             {
26.                 for (j = index; j < n; j++)
27.                 {
28.                     for (i = start; i < index; i++)
29.                         printf("%c", a[i]);
30.                     printf("%c\n", a[j]);
31.                 }
32.                 if (start != i)
33.                 {
34.                     start++;
35.                     index = start;
36.                 }
37.             }
38.             else
39.             {
40.                 index++;
41.             }
42.         }
43.     }
44. }
```

advertisement

\$ cc string20.c

\$ a.out

Enter the string

programming

enter input length

2

The subsets are

pr

po

pg

pr

pa

pm

pm

pi

pn

pg

enter the string

programming

enter input length

4

The subsets are

prog

pror

proa

prom
prom
proi
pron
prog

214.C Program to Check if nth Bit in a 32-bit Integer is Set or not

[« Prev](#)

[Next »](#)

This is a C Program to check if nth bit in a 32-bit integer is Set or not.

Problem Description

This C Program checks if nth bit in a 32-bit integer is set or not.

Problem Solution

Take input from the user and checks whether the position is set or not as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to check if nth bit in a 32-bit integer is set or not. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Check if nth Bit in a 32-bit Integer is Set or not
 */
#include <stdio.h>

/* global variables */
int result,position;
/* function prototype */
int n_bit_position(int x,int position);

void main()
{
    unsigned int number;

    printf("Enter the unsigned integer:\n");
    scanf("%d", &number);
    printf("enter position\n");
    scanf("%d", &position);
    n_bit_position(number, position);
    if (result & 1)
        printf("YES\n");
    else
        printf("NO\n");
}

/* function to check whether the position is set to 1 or not */
int n_bit_position(int number,int position)
{
    result = (number>>(position));
```

Program Explanation

In this C Program, we are reading the unsigned integer and position using ‘number’ and ‘position’ variables respectively. The n_bit_position() function is used to check whether the position is set to 1 or not.

advertisement

The result variable is used to perform Binary Right Shift Operator, the left operand’s value is moved right by the number of bits specified by the right operands.

If else condition statement is used to check that value of ‘result’ variable consists of 1 bit. If the condition is true, then execute the statement and print the output of the program.

Runtime Test Cases

```
$ cc bit32.c
$ a.out
Enter the unsigned integer:
101
```

```
enter position
4
NO

$ a.out
Enter the unsigned integer:
113
enter position
4
YES
```

215.C Program to Perform Binary Addition of Strings and Print it

[« Prev](#)

[Next »](#)

This is a C Program to perform binary addition of strings and print it.

Problem Description

This C Program Performs Binary Addition of strings and Print it.

Problem Solution

Take input from the user and performs binary addition as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Perform Binary Addition of strings and Print it. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Perform Binary Addition of Strings and Print it
 */
#include <stdio.h>
#include <string.h>

/* global variables */
char s1[10], s2[10], s3[10];
int i, k;
char carry = '0';
/* function prototype */
void binary_add(char *,char *);

void main()
{
    printf("enter string1\n");
    scanf(" %[^\n]", s1);
    printf("enter string2\n");
    scanf(" %[^\n]", s2);
    binary_add(s1, s2);
    printf("binary addition of number is\n");
    if (carry == '1')
    {
        s3[i] = '1';
        for (i = 1;i <= k + 1;i++)
            printf("%c", s3[i]);
        printf("\n");
    }
    else
    {
        for (i = 1;i <= k + 1;i++)
            printf("%c", s3[i]);
        printf("\n");
    }
}

/*function to add two binary numbers in a string */
void binary_add(char *s1, char *s2)
{
    char *p1, *p2;
    p1 = s1;
    p2 = s2;
    k = strlen(s1);

    for (;*p1 != '\0' && *p2 != '\0';p1++, p2++)
    {
        p1--;
        p2--;
        s3[k+1] = '\0';
        for (i = k + 1;i >= 1;i--, p1--, p2--)
        {
            if (*p1 == '0' && *p2 == '0' && carry == '0')
            {
                s3[i] = (*p1 ^ *p2) ^ carry;
            }
            else
                s3[i] = (*p1 ^ *p2) ^ carry ^ 1;
        }
        k--;
    }
}
```

```

    carry = '0';
}
else if (*p1 == '0' && *p2 == '0' && carry == '1')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '0';
}
else if (*p1 == '0' && *p2 == '1' && carry == '0')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '0';
}
else if (*p1 == '0' && *p2 == '1' && carry == '1')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '1';
}
else if (*p1 == '1' && *p2 == '0' && carry == '0')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '0';
}
else if (*p1 == '1' && *p2 == '0' && carry == '1')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '1';
}
else if (*p1 == '1' && *p2 == '1' && carry == '0')
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '1';
}
else
{
    s3[i] = (*p1 ^ *p2)^ carry;
    carry = '1';
}
}
}

```

Program Explanation

In this C Program, we are reading the string1 and string2 using ‘s1’ and ‘s2’ variables respectively. The binary add() function is used to add two binary numbers in a string.

advertisement

Nested if else condition statement is used to check that three variable values string1, string2 and carry variable values should be {000.001,010,011,100,101,110} using logical AND operator.

If any one of the condition is true then execute the statement. Using Binary OR Operator copy a bit if it is set in one operand but not both and assign the value to s3[] array variable. Using if else condition statement print the binary addition of strings.

Runtime Test Cases

```
$ cc bit20.c  
$ a.out  
enter string1  
00010001  
enter string2  
00010010  
binary addition of number is
```

216.C Program to Check if a given Integer is a Power of 2 without using Bitwise

[« Prev](#)

[Next »](#)

This C Program checks if a given integer is a power of 2 without using bitwise.

Here is source code of the C Program to check if a given integer is a power of 2 without using bitwise. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Check if a given Integer is a Power of 2 without using Bitwise
3. */
```

```

4. #include <stdio.h>
5.
6. /*function prototype */
7. int power_of_2(unsigned int);
8. /* gloabal variables */
9. int b[32] = {0}, j = 0, n, i, count = 0;
10.
11.void main()
12.{
13.    unsigned int num;
14.
15.    printf("enter value\n");
16.    scanf("%d", &num);
17.    power_of_2(num);
18.    if (count == 1)
19.        printf("YES\n");
20.    else
21.        printf("NO\n");
22.}
23.
24./*function to check whether a given number is power of 2 or not */
25.int power_of_2(unsigned int num)
26.{
27.    while (num != 0)
28.    {
29.        n = num % 2;
30.        if (n == 1)
31.            count++;
32.        num = num / 2;
33.    }
34.}

```

advertisement

```
$ cc bit26.c
$ a.out
enter value
128
YES
```

```
$ a.out
enter value
126
NO
```

217.C Program to Check if a given Bit Position is set to One or not

[« Prev](#)

[Next »](#)

This is a C Program to check if a given bit position is set to one or not.

Problem Description

This C Program checks if a given bit position is set to one or not.

Problem Solution

Take input from the user and checks bit position as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to check if a given bit position is set to one or not. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Check if a given Bit Position is set to One or not
 */
#include <stdio.h>

void main()
{
    unsigned int number;
    int result, position;

    printf("Enter the unsigned integer:\n");
    scanf("%d", &number);
    printf("enter position to be searched\n");
    scanf("%d", &position);
    result = (number >> (position));
    if (result & 1)
        printf("TRUE\n");
    else
        printf("FALSE\n");
}
```

Program Explanation

In this C Program, we are reading the unsigned integer and position to be searched using ‘number’ and ‘position’ variables respectively. Compute the Binary Right Shift Operation.

advertisement

The left operand value is moved right by the number of bits specified by the right operand. If else condition statement is used to copy a bit to the result if it exists in both operands using Binary AND operator. Print the bit position which is set to one or not.

Runtime Test Cases

```
$ cc bit14.c
$ a.out
Enter the unsigned integer:
128
enter position to be searched
7
TRUE
```

218.C Program to Reverse all the Bits of an 32-bit Integer using Bitwise

[« Prev](#)

[Next »](#)

This is a C Program to reverse all the bits of an 32-bit integer using bitwise.

Problem Description

This C Program reverse all the bits of an 32-bit integer using bitwise.

Problem Solution

Take input from the user and performs bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to reverse all the bits of an 32-bit integer using bitwise. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Reverse all the Bits of an 32-bit Integer using
 * Bitwise
 */
#include <stdio.h>

int main ()
{
    int n = 0, num = 0, count = 0, rev_bits = 0;
    printf ("Enter the number: ");
    scanf ("%d", &n);

    while (n > 0)
    {
        // keep shifting each bit
        rev_bits = rev_bits << 1;

        // if the bit is 1 then we XOR with 1
        if (n & 1 == 1)
        {
            rev_bits = rev_bits ^ 1;
        }

        // right shift n
        n = n >> 1;
    }

    printf ("\nThe reversed resultant = %d\n", rev_bits);

    return 0;
}
```

Program Explanation

1. Take the input from the user and store it in “n” variable.
2. For a given integer n, the basic idea is to loop through each bit of ‘n’ from right end (right-shift) and keep shifting ‘rev_bits’ from left end (left-shift).
$$\text{rev_bits} = \text{rev_bits} \ll 1; \text{n} = \text{n} \gg 1;$$
3. In while looping if a set bit is encountered, then set the bit in rev_bits. Perform loop for all bits. At the end ‘rev_bits’ contain the resultant reverse.

advertisement

Runtime Test Cases

Test case 1 – Here, the entered number is 2.

Enter the number: 2

The reversed resultant = 1

Test case 2 – Here, the entered number is 7.

advertisement

Enter the number: 7

The reversed resultant = 7

Test case 3 – Here, the entered number is 256

Enter the number: 256

The reversed resultant = 1

219.C Program to Count the Number of Bits set to One using Bitwise Operations

[« Prev](#)

[Next »](#)

This is a C Program to count the number of bits set to one using bitwise operations.

Problem Description

This C Program Counts the Number of Bits set to One using Bitwise Operations.

Problem Solution

Take input from the user and performs bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Count the Number of Bits set to One using Bitwise Operations. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Count the Number of Bits set to One using
 * Bitwise Operations
 */
#include <stdio.h>

int main()
{
    unsigned int number;
    int count = 0;

    printf("Enter the unsigned integer:\n");
    scanf("%d", &number);
    while (number != 0)
    {
        if ((number & 1) == 1)
            count++;
        number = number >> 1;
    }
    printf("number of one's are :\n%d\n", count);
    return 0;
}
```

Program Explanation

In this C Program, we are reading the unsigned integer using ‘number’ variable. Using while loop count the number of bits set to one using bitwise operations.

advertisement

If condition statement is used to check the value which copies a bit to the result if it exists in both operands using Binary AND operator is equal to 1. If the condition is true, then execute the statement and increment the number of bits set to 1 using count variable.

Using Binary Right Shift Operator, the left operand’s value is moved right by the number of bits specified by the right operands and assign the value to ‘number’ variable. Hence we are displaying the output of the program. Print the count of the number of bits set to one using bitwise operations.

Runtime Test Cases

```
$ cc bit2.c
$ a.out
Enter the unsigned integer:
128
number of one's are :
1
```

```
$ a.out
Enter the unsigned integer:
-127
number of one's are :
26
```

220.C Program to Update Details of Employee using Files

[« Prev](#)

[Next »](#)

This C Program Updates the Details of Employee using Files.

Here is source code of the C Program to Update Details of Employee using Files. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Update Details of Employee using Files
3.  */
4. #include <stdio.h>
```

```
5. #include <stdlib.h>
6. #include <string.h>
7. struct emp
8. {
9.     int empid;
10.    char *name;
11.};
12.
13.int count = 0;
14.void add_rec(char *a);
15.void display(char *a);
16.void update_rec(char *a);
17.
18.void main(int argc, char *argv[])
19.{
20.    int choice;
21.    while (1)
22.    {
23.        printf("MENU:\n");
24.        printf("1.Add a record\n");
25.        printf("2.Display the file\n");
26.        printf("3.Update the record\n");
27.        printf("Enter your choice:");
28.        scanf("%d", &choice);
29.
30.        switch(choice)
31.        {
32.            case 1:
33.                add_rec(argv[1]);
34.                break;
35.            case 2:
36.                display(argv[1]);
37.                break;
38.            case 3:
39.                update_rec(argv[1]);
40.                break;
41.            case 4:
42.                exit(0);
43.            default:
44.                printf("Wrong choice!!!\nEnter the correct choice\n");
45.        }
46.    }
47.}
48.
49.void add_rec(char *a)
50.{
51.    FILE *fp;
52.    fp = fopen(a, "a+");
53.    struct emp *temp = (struct emp *)malloc(sizeof(struct emp));
54.    temp->name = (char *)malloc(50*sizeof(char));
55.    if (fp == NULL)
56.        printf("Error!!!\n");
57.    else
58.    {
59.        printf("Enter the employee id\n");
60.        scanf("%d", &temp->empid);
61.        fwrite(&temp->empid, sizeof(int), 1, fp);
62.        printf("enter the employee name\n");
63.        scanf(" %[^\n]s", temp->name);
64.        fwrite(temp->name, 50, 1, fp);
65.        count++;
66.    }
67.    fclose(fp);
68.    free(temp);
69.    free(temp->name);
70.}
```

```

71.
72.void display(char *a)
73.{ 
74.    FILE *fp;
75.    char ch;
76.    int rec = count;
77.    fp = fopen(a, "r");
78.    struct emp *temp = (struct emp *)malloc(sizeof(struct emp));
79.    temp->name = (char *)malloc(50*sizeof(char));
80.    if (fp == NULL)
81.        printf("Error!!!");
82.    else
83.    {
84.        while (rec)
85.        {
86.            fread(&temp->empid, sizeof(int), 1, fp);
87.            printf("%d", temp->empid);
88.            fread(temp->name, 50, 1, fp);
89.            printf(" %s\n", temp->name);
90.            rec--;
91.        }
92.    }
93.    fclose(fp);
94.    free(temp);
95.    free(temp->name);
96.}
97.
98.void update_rec(char *a)
99.{ 
100.    FILE *fp;
101.    char ch, name[5];
102.    int rec, id, c;
103.    fp = fopen(a, "r+");
104.    struct emp *temp = (struct emp *)malloc(sizeof(struct emp));
105.    temp->name = (char *)malloc(50*sizeof(char));
106.    printf("Enter the employee id to update:\n");
107.    scanf("%d", &id);
108.    fseek(fp, 0, 0);
109.    rec = count;
110.    while (rec)
111.    {
112.        fread(&temp->empid, sizeof(int), 1, fp);
113.        printf("%d", temp->empid);
114.        if (id == temp->empid)
115.        {
116.            printf("Enter the employee name to be updated");
117.            scanf(" %[^\n]s", name);
118.            c = fwrite(name, 50, 1, fp);
119.            break;
120.        }
121.        fread(temp->name, 50, 1, fp);
122.        rec--;
123.    }
124.    if (c == 1)
125.        printf("Record updated\n");
126.    else
127.        printf("Update not successful\n");
128.    fclose(fp);
129.    free(temp);
130.    free(temp->name);
131.}

```

advertisement

```

$ cc file5.c
$ a.out empl
MENU:

```

```
1.Add a record
2.Display the file
3.Update the record
Enter your choice:1
Enter the employee id
1
enter the employee name
aaa
MENU:
1.Add a record
2.Display the file
3.Update the record
Enter your choice:1
Enter the employee id
2
enter the employee name
bbb
MENU:
1.Add a record
2.Display the file
3.Update the record
Enter your choice:3
Enter the employee id to update:
1
1Enter the employee name to be updated1bc
Record updated
MENU:
1.Add a record
2.Display the file
3.Update the record
Enter your choice:2
1 1bc
2 bbb
MENU:
1.Add a record
2.Display the file
3.Update the record
Enter your choice:4
```

221.C Program To Print Smallest and Biggest possible Word which is Palindrome in a given String

[« Prev](#)

[Next »](#)

This is a C program to print smallest and biggest possible word which is palindrome in a given string.

Problem Description

This C Program print smallest and biggest possible word which is palindrome in a given string.

Problem Solution

Take input from the user and print smallest and biggest possible word which is palindrome in a given string as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to print smallest and biggest possible word which is palindrome in a given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program To Print Smallest and Biggest possible Word
 * which is Palindrome in a given String
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    int i = 0, l = 0, j, k, space = 0, count = 0, init = 0, min = 0, max = 0, len = 0, flag;
    char a[100], b[30][20], c[30], d[30], minP[30], maxP[30];

    printf("Read a string:\n");
    fflush(stdin);
    scanf("%[^\\n]s", a);
    for (i = 0; a[i] != '\0'; i++)
    {
        if (a[i] == ' ')
            space++;
    }
    i = 0;
    for (j = 0; j < (space + 1); i++, j++)
    {
        k = 0;
        while (a[i] != '\0')
        {
            if (a[i] == ' ')
            {
                break;
            }
            else
            {
                b[j][k++] = a[i];
                i++;
            }
        }
        b[j][k] = '\0';
    }
    for (j = 0; j < space + 1; j++)
        printf("%os ", b[j]);
    printf("\n");
    for (i = 0; i < space + 1; i++)
    {
        strcpy(c, b[i]);
```

```

count = strlen(b[i]);
k = 0;
for (l = count - 1;l >= 0;l--)
    d[k++] = b[i][l];
d[k] = '\0';
if(strcmp(d, c) == 0)
    flag = 1;
if(init < 1)
{
    strcpy(minP, d);
    strcpy(maxP, d);
    min = strlen(minP);
    max = strlen(maxP);
    init++;
}
printf("String %s is a Palindrome\n", d);
len = strlen(d);
if(len >= max)
    strcpy(maxP, d);
else if(len <= min)
    strcpy(minP, d);
else
    printf("");
}
if(flag == 1)
{
    printf("The minimum palindrome is %s\n", minP);
    printf("The maximum palindrome is %s\n", maxP);
}
else
    printf("given string has no pallindrome\n");
}

```

Program Explanation

In this C program, we are reading the string to ‘a’ character[] array variable. For loop is used to count the number of space present in between the words. Another for loop is used to assign the string from the value of ‘a’ character variable to b[] character variable. Using while loop check the value of a[i] character variable value is not equal to null. If the condition is true then execute the while loop.

advertisement

If-else condition statement is used to check the value of ‘character’ variable is equal to empty space. If the condition is true then execute the statement, break command is used to stop iteration of the loop. Otherwise, if the condition is false, then execute the else statement. Assigning the value of a[i] character variable to b[] character variable.

For loop is used to find the smallest and biggest possible words which is Palindrome in a given String. The strcpy() function is used copy the b[] array variable value to ‘c’ variable. Using ‘count’ variable compute the length of the string in b[] array variable.

In another for loop initialize the value of ‘l’ variable as the difference between the values of ‘count’ variable by 1. Check the condition that the value of ‘l’ variable is greater than or equal to 0. Using if condition statement check the strcmp() function value is equal to 0. If the condition is true then execute the statement.

advertisement

Another if condition statement is used to check the value of ‘init’ variable is less than 1. Strcpy() function is used to copy the value of ‘d’ variable to ‘minp’ variable and to ‘maxp’ variable. The ‘min’ variable is used to compute the length of the value of ‘minp’ variable. The ‘max’ variable is used to compute the length of the value of ‘maxp’ variable.

Nested if else condition statement is used to find the smallest and biggest possible word palindrome in a given string. Check the value of ‘len’ variable is greater than the value of ‘max’ variable. If the condition is true then execute the statement. Copy the value of ‘d’ string variable to ‘maxp’ variable. Otherwise, if the condition is false, then execute the else if statement. Check the value of ‘len’ variable is less than or equal to the value of ‘min’ variable.

If the condition is true, then execute the statement. Copy the value of ‘d’ string variable to ‘minp’ variable. If the value of ‘flag’ variable is equal to 1 then print the statement as the minimum and maximum strings in the palindrome. Otherwise, if the condition is false, then execute the else statement and print the statement as the string is not palindrome.

advertisement

Runtime Test Cases

```
$ cc string14i.c
$ a.out
Read a string:
aba abcba abcdcba bcd
aba abcba abcdcba bcd
String aba is a Palindrome
String abcba is a Palindrome
String abcdcba is a Palindrome
The minimum palindrome is aba
The maximum palindrome is abcdcba

$ a.out
Read a string:
abc abcd
abc abcd
given string has no pallindrome
```

222.C Program to Count Number of Words in a given Text or Sentence

[« Prev](#)

[Next »](#)

This is a C Program to Count the Number of Words in a given text or Sentence.

Problem Description

This program takes a string as input and count the number of words in the input string.

Problem Solution

1. Take a string as input.
2. Using for loop search for a empty space in between the words in the string.
3. Consecutively increment a variable. This variable gives the count of number of words.

advertisement

Program/Source Code

Here is source code of the C Program to Count the Number of Words in a given text Or Sentence. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Count Number of Words in a given Text Or Sentence
3. */
4. #include <stdio.h>
5. #include <string.h>
6.
7. void main()
8. {
9.     char s[200];
10.    int count = 0, i;
11.
12.    printf("Enter the string:\n");
13.    scanf("%[^\\n]s", s);
14.    for (i = 0; s[i] != '\\0'; i++)
15.    {
16.        if (s[i] == ' ' && s[i+1] != ' ')
17.            count++;
18.    }
19.    printf("Number of words in given string are: %d\n", count + 1);
20.}
```

Program Explanation

1. Take a string as input and store it in the array s[].
2. Using for loop search for a space ‘ ‘ in the string and consecutively increment a variable count.
3. Do step-2 until the end of the string.
4. Increment the variable count by 1 and then print the variable count as output.

advertisement

Runtime Test Cases

```
Enter the string:  
welcome to sanfoundry's c-programming class!  
Number of words in given string are: 5
```

```
Enter the string:  
Best Reference Books in C Programming  
Number of words in given string are: 6
```

223.C Program to Delete All Repeated Words in String

« [Prev](#)

[Next](#) »

This C Program Deletes All Repeated Words in String.

Here is source code of the C Program to Delete All Repeated Words in String. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Delete All Repeated Words in String
3. */
4. #include <stdio.h>
```

```

5. #include <stdlib.h>
6. #include <string.h>
7.
8. int main ()
9. {
10.     char str[100], word[100], twoD[10][30];
11.     int i = 0, j = 0, k = 0, len1 = 0, len2 = 0, l = 0;
12.
13.     printf ("Enter the string\n");
14.     gets (str);
15.
16. // let us convert the string into 2D array
17. for (i = 0; str[i] != '\0'; i++)
18. {
19.     if (str[i] == ' ')
20.     {
21.         twoD[k][j] = '\0';
22.         k++;
23.         j = 0;
24.     }
25.     else
26.     {
27.         twoD[k][j] = str[i];
28.         j++;
29.     }
30. }
31.
32. twoD[k][j] = '\0';
33.
34. j = 0;
35. for (i = 0; i < k; i++)
36. {
37.     int present = 0;
38.     for (l = 1; l < k + 1; l++)
39.     {
40.         if (twoD[l][j] == '\0' || l == i)
41.         {
42.             continue;
43.         }
44.
45.         if (strcmp (twoD[i], twoD[l]) == 0) {
46.             twoD[l][j] = '\0';
47.             present = present + 1;
48.         }
49.     }
50. // if (present > 0)      | uncomment this `if` block if you
51. // {                      | want to remove all the occurrences
52. //     twoD[i][j] = '\0'; | of the words including the word
53. // }                      | itself.
54. }
55.
56. j = 0;
57.
58. for (i = 0; i < k + 1; i++)
59. {
60.     if (twoD[i][j] == '\0')
61.         continue;
62.     else
63.         printf ("%s ", twoD[i]);
64. }
65.
66. printf ("\n");
67.
68. return 0;
69. }
```

Enter the string
welcome to sanfoundry's c programming class , welcome again to c class !
welcome to sanfoundry's c programming class , again !

Enter the string:
Welcome to Sanfoundry C Class, Welcome to Java Programming, Welcome to C++ class
Welcome to Sanfoundry C Class, Java Programming, C++ class

224.C Program to Find the Frequency of Every Word in a given String

[« Prev](#)

[Next »](#)

This C Program finds frequency of every word in a given string.

Here is source code of the C Program to find the frequency of every word in a given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Frequency of Every Word in a
3. * given String
4. */
5. #include <stdio.h>
```

```
6. #include <string.h>
7.
8. void main()
9. {
10.    int count = 0, c = 0, i, j = 0, k, space = 0;
11.
12.    char str[100], p[50][100], str1[20], ptr1[50][100];
13.
14.    char *ptr;
15.
16.    printf("Enter the string\n");
17.    scanf(" %[^\n]s", str);
18.
19.    printf("string length is %d\n", strlen(str));
20.
21.    for (i = 0;i<strlen(str);i++)
22.    {
23.        if ((str[i] == ' ')||(str[i] == ',' && str[i+1] == ',')||(str[i] == '.'))
24.        {
25.            space++;
26.        }
27.    }
28.
29.    for (i = 0, j = 0, k = 0;j < strlen(str);j++)
30.    {
31.        if ((str[j] == ' ')||(str[j] == 44)|| (str[j] == 46))
32.        {
33.            p[i][k] = '\0';
34.            i++;
35.            k = 0;
36.        }
37.        else
38.            p[i][k++] = str[j];
39.    }
40.
41.    k = 0;
42.
43.    for (i = 0;i <= space;i++)
44.    {
45.        for (j = 0;j <= space;j++)
46.        {
47.            if (i == j)
48.            {
49.                strcpy(ptr1[k], p[i]);
50.                k++;
51.                count++;
52.
53.                break;
54.            }
55.            else
56.            {
57.                if (strcmp(ptr1[j], p[i]) != 0)
58.                    continue;
59.                else
60.                    break;
61.            }
62.        }
63.    }
64.
65.    for (i = 0;i < count;i++)
66.    {
67.        for (j = 0;j <= space;j++)
68.        {
69.            if (strcmp(ptr1[i], p[j]) == 0)
70.                c++;
71.        }
72.    }
73.
```

```
72.     printf("%s -> %d times\n", ptr1[i], c);
73.     c = 0;
74. }
75.}
```

advertisement

Enter the string:

Welcome to Sanfoundry Welcome to C Class

string length is 40

Welcome -> 2 **times**

to -> 2 **times**

Sanfoundry -> 1 **times**

C -> 1 **times**

Class -> 1 **times**

Enter the string:

Welcome to C Class, Java Class

string length is 30

Welcome -> 1 **times**

to -> 1 **times**

C -> 1 **times**

Class -> 2 **times**

-> 1 **times**

Java -> 1 **times**

225.C Program to Find the Frequency of Substring in the given String

[« Prev](#)

[Next »](#)

This is a C Program to find the frequency of substring in the given string.

Problem Description

This program finds the frequency of substring in the given string.

Problem Solution

1. Take a string and a substring as input.
2. Compare the substring with the main string.
3. Count the number of times it matches in the main string and print the count as output.

advertisement

Program/Source Code

Here is source code of the C Program to find the frequency of substring in the given string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Frequency of Substring in
3. * the given String
4. */
5. #include <stdio.h>
6. #include <string.h>
7.
8. void main()
9. {
10.    int count = 0, i, j = 0, k;
11.    char str[100], str1[20];
12.
13.    printf("Enter the string:\n");
14.    scanf(" %[^\n]s", str);
15.
16.    printf("Enter the substring to be matched:\n");
17.    scanf(" %[^\n]s", str1);
18.
19.    k = strlen(str1);
20.
21.    for (i = 0; str[i] != '\0'; i++)
22.    {
23.        while (str[i] == str1[j])
24.        {
25.            j++;
26.        }
27.
28.        if (j == k)
29.        {
30.            count++;
31.            j = 0;
32.        }
33.    }
34.    printf("No of matches of substring in main string is: %d\n", count);
35.}
```

Program Explanation

1. Take a string and a substring as input and store it in the arrays str[] and str1[] respectively.
2. Using for loop compare str1[] with the str[].
3. Do step-2 until the end of the main string.
4. During the comparison increment the variable count whenever the substring matches in the main string.
5. Print the variable count as output.

advertisement

Runtime Test Cases

Enter the string:
prrogramm is prrogramming

```
Enter the substring to be matched:  
rr  
No of matches of substring in main string is: 4
```

```
Enter the string:  
Sanfoundry C Programming  
Enter the substring to be matched:  
oun  
No of matches of substring in main string is: 1
```

226.C Program to Identify the Missing Number in an Integer Array of Size N-1 with Numbers[1,N]

[« Prev](#)

[Next »](#)

This is a C Program to identify the missing number in an integer array of size n-1 with numbers[1,N].

Problem Description

This C Program Identifies the Missing Number in an Integer Array of Size N-1 with Numbers[1,N].

Problem Solution

Take input from the user and performs bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to To Identify the Missing Number in an Integer Array of Size N-1 with Numbers[1,N]. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program To Identify the Missing Number in an Integer
 * Array of Size N-1 with Numbers[1,N]
 */
#include <stdio.h>
#define MAX 15
int missing_number_array(int [],int);

int main()
{
    int a[MAX], num, i, n;

    printf("enter the range of array\n");
    scanf("%d", &n);
    for (i = 0;i < n;i++)
    {
        printf("enter a[%d]element into the array:", i);
        scanf("%d", &a[i]);
    }
    num = missing_number_array(a, n);
    printf("The missing number -> %d\n", num);
}

/* To find the missing number in array */
int missing_number_array(int a[], int n)
{
    int i;
    int s1 = 0;
    int s2 = 0;

    for (i = 0;i < n;i++)
        s1 = s1 ^ a[i];
    for (i = 1;i <= n + 1;i++)
        s2 = s2 ^ i;
    return (s1 ^ s2);
}
```

Program Explanation

In this C Program, we are reading the range of array using ‘n’ variable. Using for loop we are entering the coefficient element values of an array. The `missing_number_array()` function is used to find the missing number in array.

advertisement

Then ‘s1’ and ‘s2’ variables are used to compute the Binary XOR operation, copies the bit if it is set in one operand but not both. Again compute the Binary XOR Operation is performed for s1 and s2 variable value and returns the value. Print the missing number in an integer array of size N-1 with numbers [1, N].

Runtime Test Cases

```
$ cc bit29.c
$ a.out
enter the range of array
9
enter a[0]element into the array:1
enter a[1]element into the array:5
enter a[2]element into the array:2
enter a[3]element into the array:7
enter a[4]element into the array:3
enter a[5]element into the array:4
enter a[6]element into the array:10
enter a[7]element into the array:9
enter a[8]element into the array:6
The missing number -> 8
$ a.out
enter the range of array
4
enter a[0]element into the array:1
enter a[1]element into the array:5
enter a[2]element into the array:3
enter a[3]element into the array:2
The missing number -> 4
$ a.out
enter the range of array
4
enter a[0]element into the array:3
enter a[1]element into the array:2
enter a[2]element into the array:5
enter a[3]element into the array:4
The missing number -> 1
```

227.C Program to Use Bitwise Operations to Count the Number of Leading Zero's in a Number x

[« Prev](#)

[Next »](#)

This is a C Program to use bitwise operations to count the number of leading zero's in a number x.

Problem Description

This C Program uses Bitwise operations to count the number of leading zero's in a number x.

Problem Solution

Take input from the user and perform bitwise operations as shown in the program below.

Program/Source Code

Here is source code of the C Program to use Bitwise operations to count the number of leading zero's in a number x. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Use Bitwise Operations to Count the Number of
 * Leading Zero's in a Number x
 */
#include <stdio.h>
#include <malloc.h>

int main()
{
    int lim = sizeof(int) * 8;
    // the mBit is 1000 0000 0000 0000 0000 0000 0000 0000
    int mBit = 1 << (lim - 1);
    int num = 0, count = 0;
    printf ("Enter the digit: ");
    scanf ("%d", &num);
    while (!(num & mBit))
    {
        num = (num << 1);
        count++;
    }
    printf ("\nNumber of leading zero's is: %d\n", count);
    return 0;
}
```

Program Explanation

1. Take the input from the user and store it in the num variable. Leading zero's in a binary number is equal to zeros preceding the highest order set bit of the number.
2. int lim = sizeof(int) * 8; is used to find the total number of bits required to store an integer in memory.
3. For Setting MSG, initialize a variable and set its MSB to 1. We can achieve this by using int mBit = 1 << (lim - 1); So the mBit will be 10000000 00000000 00000000 00000000. It stores the number of leading zero's in the variable count.
4. While statement is used to find the leading set bit. If the leading set bit is found we terminate the loop, else right shift the bit by 1 and we increase the value of count by 1.
5. Print the number of leading zero's in the given number using print statement.

Runtime Test Cases

Test case 1 – Here, the entered digit is 16.

```
gcc leading_zeros.c -o leading-zero
./leading-zero
```

Enter the digit: 16
Number of leading zero's is: 27

Test case 2 – Here, the entered digit is 64.

```
gcc leading_zeros.c -o leading-zero
./leading-zero
```

```
Enter the digit: 64
Number of leading zero's is: 25
Test case 3 – Here, the entered digit is 1.
```

```
gcc leading_zeros.c -o leading-zero
./leading-zero
```

```
Enter the digit: 1
Number of leading zero's is: 31
```

228.C Program to Count the Number of Bits needed to be Flipped to Integer X to Generate Integer Y

[« Prev](#)

[Next »](#)

This is a C Program to count the number of bits needed to be flipped to integer X to generate integer Y.

Problem Description

This C Program counts the number of bits needed to be flipped to integer X to generate integer Y.

Problem Solution

Take input from the user and performs binary right shift operation as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to count the number of bits needed to be flipped to integer X to generate integer Y. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Count the Number of Bits needed to be Flipped
 * to Integer X to Generate Integer Y
 */
#include <stdio.h>
#include <stdlib.h>
#define NUM_BITS_INT (sizeof(int)*8)

void main()
{
    int n, m, i, count = 0, a, b;

    printf("Enter the number\n");
    scanf("%d", &n);
    printf("Enter another number\n");
    scanf("%d", &m);
    for (i = NUM_BITS_INT-1; i >= 0; i--) {
        a = (n >> i) & 1;
        b = (m >> i) & 1;
        if (a != b)
            count++;
    }
    printf("flip count = %d\n", count);
}
```

Program Explanation

In this C Program, we are reading the two integer values using ‘n’ and ‘m’ variables respectively. For loop is used to count the number of bits needed to be flipped to integer X to generate integer Y.

advertisement

Binary Right shift operation, the left operand’s value is moved right by the number of bits specified by the right operands then copy a bit to the result if it exists in both operands.

If condition statement is used to check that the value of bit is not equal. If the condition is true, then execute the statement and print the number of bits to be flipped to integer X to generate integer Y.

Runtime Test Cases

```
$ cc flip.c
$ a.out
Enter the number
127
Enter another number
125
flip count = 1
```

```
$ a.out
Enter the number
127
Enter another number
128
flip count = 8
$ a.out
Enter the number
42
Enter another number
21
flip count = 6
```

229.C Program to Print Combination of two Words of two given Strings without any Repetition

[« Prev](#)

[Next »](#)

This is a C Program to print combination of two words of two given strings without any repetition.

Problem Description

This C Program prints combination of two words of two given strings without any repetition.

Problem Solution

Take input from the user and perform string operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to print combination of two words of two given strings without any repetition. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Print Combination of two Words of two
 * given Strings without any Repetition
 */
#include <stdio.h>
#include <string.h>

void main()
{
    char string[100], str[10], c[10];
    int z, occ = 0, i = 0, j = 0, count = 0, len = 0;

    printf("Enter a string:");
    scanf("%[^\\n]s", string);
    printf("Enter the word to check its occurrence:");
    scanf("%s", str);
    len = strlen(str);
    for (i = 0; string[i] != '\0'; i++)
    {
        count = 0;
        for (j = 0, z = i; j < len; j++, z++)
        {
            c[j] = string[z];
            if (c[j] == str[j])
            {
                count++; /* Incrementing the count if the characters of the main string match with the characters of the given word */
            }
        }
        if (count == len && string[z] == ' ')
        {
            occ++; /* Incrementing the occ if word matches completely and next character in string is space */
        }
    }
    printf("The number of occ is %d\n", occ);
}
```

Program Explanation

In this C program, we are reading a value of string using ‘string’ variable. Compute the length of the string using `strlen()` function for ‘str’ variable. For loop is used to find the combination of two words of two given strings without any repetition.

advertisement

In for loop initialize the value of ‘i’ variable as 0. Check the condition that the value of ‘string[]’ array variable with base index of the value of ‘i’ variable is not equal to null. If the condition is true then execute the iteration of the loop, initialize the value of ‘count’ variable to 0.

In another for loop initialize the value of ‘j’ variable to 0 and the value of ‘z’ variable to the value of ‘i’ variable. Check the condition that the value of ‘j’ variable is less than the value of ‘len’ variable. If the condition is true

then execute the loop. If condition statement is used to check that the characters in the main string match with the characters of the given word.

If the condition is true then execute the statement and increment the value of ‘count’ variable. Another if condition statement is used to check that word matches completely and next character in a string is space is not to be true using logical AND operator. If the condition is true then execute the statement and increment the value of ‘occ’ variable. Print combination of two words of two given strings without any repetition.

advertisement

Runtime Test Cases

```
$ cc string3.c
$ a.out
Enter a string:welcome to sanfoundry's c programming class, welcome again to c class
Enter the word to check its occurrence:welcome
The number of occ is 2
```

```
$ cc string3.c
$ a.out
Enter a string:welcome to sanfoundry's c programming class, welcome again to c class
Enter the word to check its occurrence:c
The number of occ is 2
```

230.C Program to Collect Statistics of a Source File like Total Lines, Total no. of Blank Lines, Total no. of Lines ending with Semicolon

« [Prev](#)

[Next](#) »

This C Program Collect Statistics of a Source File like Total Lines, Total no. of Blank Lines, Total no. of Lines ending with Semicolon.

Here is source code of the C Program to Collect Statistics of a Source File like Total Lines, Total no. of Blank Lines, Total no. of Lines ending with Semicolon. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Collect Statistics of a Source File like Total Lines,
```

```

3. * Total no. of Blank Lines, Total no. of Lines ending with Semicolon
4. */
5. #include <stdio.h>
6. #include <stdlib.h>
7.
8. void main(int argc, char *argv[]) /* Command line Arguments */
9. {
10.    int ncount = 0, ccount = 0, scount = 0, blank = 0;
11.    char ch;
12.    FILE *fp;
13.    fp = fopen(argv[1], "r");
14.    if (fp == NULL)
15.    {
16.        perror("Error Occured");
17.    }
18.    else
19.    {
20.        while(1)
21.        {
22.            ch = fgetc(fp);
23.            if (ch == EOF)
24.            {
25.                break;
26.            }
27.            if(ch == 10)
28.            {
29.                ncount++;
30.                if (ch = fgetc(fp) == '\n')
31.                {
32.                    fseek(fp, -1, 1); /* shifting offset of the file to previous position */
33.                    blank++;
34.                }
35.            }
36.            else if (ch == 59)
37.            {
38.                scount++;
39.            }
40.            else if (ch == '/' || ch == '*')
41.            {
42.                ccount++;
43.            }
44.        }
45.    }
46.    printf("\nThe Total number of lines are %d", ncount);
47.    printf("\nThe Total number of Commented lines are %d", ccount);
48.    printf("\nThe Total number of blank lines are %d", blank);
49.    printf("\nThe total number of lines that end with Semicolon %d", scount);
50.    printf("\nThe length of Actual code is %d ", ncount-blank-ccount);
51.    fclose(fp);
52.}

```

advertisement

```
$ cc file8.c
$ a.out lines.c
```

The Total number of lines are 23
The Total number of Commented lines are 6
The Total number of blank lines are 4
The total number of lines that end with Semicolon 6
The length of Actual code is 13

231.C Program to Join Lines of Two given Files and Store them in a New file

[« Prev](#)

[Next »](#)

This C Program join lines of two given files and store them in a new file.

Here is source code of the C Program to join lines of two given files and store them in a new file. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Join Lines of Two given Files and
3.  * Store them in a New file
4. */
5. #include <stdio.h>
6. #include <stdlib.h>
7.
8. /* Function Prototype */
9. int joinfiles(FILE *, FILE *, FILE *);
```

```
10.
11. char ch;
12. int flag;
13.
14. void main(int argc, char *argv[])
15. {
16.     FILE *file1, *file2, *target;
17.
18.     file1 = fopen(argv[1], "r");
19.     if (file1 == NULL)
20.     {
21.         perror("Error Occured!");
22.     }
23.     file2 = fopen(argv[2], "r");
24.     if (file2 == NULL)
25.     {
26.         perror("Error Occured!");
27.     }
28.     target = fopen(argv[3], "a");
29.     if (target == NULL)
30.     {
31.         perror("Error Occured!");
32.     }
33.
34.     joinfiles(file1, file2, target);      /* Calling Function */
35.
36.     if (flag == 1)
37.     {
38.         printf("The files have been successfully concatenated\n");
39.     }
40. }
41.
42./* Code join the two given files line by line into a new file */
43.
44. int joinfiles(FILE *file1, FILE *file2, FILE *target)
45. {
46.     while ((fgetc(file1) != EOF) || (fgetc(file2) != EOF))
47.     {
48.         fseek(file1, -1, 1);
49.         while ((ch = fgetc(file1)) != '\n')
50.         {
51.             if (ch == EOF)
52.             {
53.                 break;
54.             }
55.             else
56.             {
57.                 fputc(ch, target);
58.             }
59.         }
60.         while ((ch = fgetc(file2)) != '\n')
61.         {
62.             if (ch == EOF)
63.             {
64.                 break;
65.             }
66.             else
67.             {
68.                 fputc(ch, target);
69.             }
70.         }
71.         fputc('\n', target);
72.     }
73.     fclose(file1);
74.     fclose(file2);
75.     fclose(target);
}
```

```
76. return flag = 1;
77. }
```

advertisement

```
$ cc file7.c
$ ./a.out lines.c words.c final.c
The files have been successfully concatenated
```

```
/* FIRST FILE */

/*
Hello!!
This is a C Program File.
Consider a code to Add two numbers
*/

#include <stdio.h>
/* Function Prototype */
int sum(int, int);
void main()
{
    int num1, num2;
    printf("Enter Number1 and Number2:");
    scanf("%d %d", &num1, &num2);
    sum(num1, num2);
}

int sum(int a, int b)
{
    return a + b;
}

/* SECOND FILE */

/*
 * this is temporary file for use in file handling
 */
#include <stdio.h>

int sqrt(int);
void main()
{
    int num;
    printf("enter the number:");
    scanf("%d", &num);
    sqrt(num);
    printf("The square of the given number is:", num);
}
int sqrt(int num)
{
    return num*num;
}

/* CONCATENATED FILE */
/*
Hello!! * this is temporary file for use in file handling
This is a C Program File. *
Consider a code to Add two numbers */
#include <stdio.h>
#include <stdio.h>
int sqrt(int);
/* Function Prototype */
void main()
{
    int sum(int, int);    int num;
    void main()    printf("enter the number:");


```

```

{   scanf("%d", &num);
    int num1, num2;  sqrt(num);
    printf("Enter Number1 and Number2:");  printf("The square of the given number is:", num);
    scanf("%d %d ", num1, num2);}
    sum(num1, num2);int sqrt(int num)
}{

    return num*num;
int sum(int a, int b)
{
    return a + b;
}

```

232.C Program to Perform Shell Sort without using Recursion

[« Prev](#)

[Next »](#)

This C Program Performs Shell Sort without using Recursion.

Here is source code of the C Program to Perform Shell Sort without using Recursion. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2. * C Program to Perform Shell Sort without using Recursion
3. */
4. #include <stdio.h>
5. #define size 7
6.

```

```

7. /* Function Prototype */
8. int shell_sort(int []);
9.
10.void main()
11.{
12.    int arr[size], i;
13.    printf("Enter the elements to be sorted:");
14.    for (i = 0;i < size;i++)
15.    {
16.        scanf("%d", &arr[i]);
17.    }
18.    shell_sort(arr);
19.    printf("The array after sorting is:");
20.    for (i = 0;i < size;i++)
21.    {
22.        printf("\n%d", arr[i]);
23.    }
24.}
25.
26./* Code to sort array using shell sort */
27.int shell_sort(int array[])
28.{
29.    int i = 0, j = 0, k = 0, mid = 0;
30.    for (k = size / 2;k > 0;k /= 2)
31.    {
32.        for (j = k;j < size;j++)
33.        {
34.            for (i = j - k;i >= 0;i -= k)
35.            {
36.                if (array[i + k] >= array[i])
37.                {
38.                    break;
39.                }
40.                else
41.                {
42.                    mid = array[i];
43.                    array[i] = array[i + k];
44.                    array[i + k] = mid;
45.                }
46.            }
47.        }
48.    }
49.    return 0;
50.}

```

advertisement

```

$ cc shellsort.c
Average case:
$ a.out
Enter the elements to be sorted:57
67
48
93
42
84
95
The array after sorting is:
42
48
57
67
84
93
95

```

Best case:

```
$ a.out
```

```
Enter the elements of array:22
```

```
33
```

```
74
```

```
85
```

```
86
```

```
87
```

```
98
```

```
The array after sorting is:22
```

```
33
```

```
74
```

```
85
```

```
86
```

```
87
```

```
98
```

Worst case:

```
$ a.out
```

```
Enter the elements of array:94
```

```
92
```

```
91
```

```
89
```

```
85
```

```
80
```

```
43
```

```
The array after sorting is:43
```

```
80
```

```
85
```

```
89
```

```
91
```

```
92
```

```
94
```

233.C Program to Check whether the given Number is Palindrome or not using Bitwise Operator

[« Prev](#)

[Next »](#)

This is a C Program to check whether the given number is palindrome or not using bitwise operator.

Problem Description

This C Program Checks whether the given Number is Palindrome or not using Bitwise Operator.

Problem Solution

Take input from the user and performs bitwise operations as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Check whether the given Number is Palindrome or not using Bitwise Operator. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Check whether the given Number is Palindrome
 * or not using Bitwise Operator
 */
#include <stdio.h>
#include <string.h>
#define SIZE 8
/* Function Prototype */
int is_palindrome(unsigned char[]);

void main()
{
    int num, num1 = 0, i = 0, j = SIZE - 1, res;
    unsigned char c[SIZE];
    printf("Enter a number(max 255)");
    scanf("%d", &num);
    num1 = num;
    while (num != 0)
    {
        c[j] = num&1;
        j--;
        num = num>>1; /* Shifting right the given number by 1 bit */
    }
    printf("The number %d in binary is:", num1);
    for (i = 0;i < SIZE;i++)
    {
        printf("%d", c[i]);
    }
    res = is_palindrome(c); /* Calling Function */
    if (res == 0)
    {
        printf("\nNUMBER IS PALINDROME\n");
    }
    else
    {
        printf("\nNUMBER IS NOT PALINDROME\n");
    }
}

/* Code to check if the number is palindrome or not */
int is_palindrome(unsigned char c[])
{
    char temp[SIZE];
    int i, j, flag = 0;
    for (i = 0, j = SIZE - 1;i < SIZE, j >= 0;i++, j--)
    {
        temp[j] = c[i];
    }
    for (i = 0;i < SIZE;i++)
    {
        if (temp[i] != c[i])
        {
```

```
    flag = 1;
}
return flag;
}
```

Program Explanation

In this C program, we are reading the value of ‘string’ using word character array variable. A palindrome is a word, phrase or sentence that reads the same backward or forward.

advertisement

Check the values of a given string and reverse string are equal. If else condition statement is used inside for loop if the condition is true then assign the value of ‘flag’ variable as 1. Otherwise, if the condition is false then execute the else statement. Assign the value of ‘flag’ variable as 0 and exit the loop.

If else condition statement is used to check the value of ‘flag’ variable is equal to 1. If the condition is true then execute the statement and print the string is palindrome. Otherwise, if the condition is false then execute the else statement and print as not a palindrome.

Runtime Test Cases

```
$ cc bits21.c
$ a.out
Enter a number(max 255)153
The number 153 in binary is:10011001
NUMBER IS PALINDROME

$ a.out
Enter a number(max 255)24
The number 24 in binary is:00011000
NUMBER IS PALINDROME
```

234.C Program to Print the Program Name and All its Arguments

[« Prev](#)

[Next »](#)

This is a C Program to print the program name and all its arguments.

Problem Description

This C Program Prints the Program Name and All its Arguments.

Problem Solution

It prints the program name and its Arguments using command line argument as shown in the program below.

advertisement

Program/Source Code

Here is source code of the C Program to Print the Program Name and All its Arguments. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Print the Program Name and All its Arguments
 */
#include <stdio.h>

void main(int argc, char *argv[]) /* command line Arguments */
{
    int i;
    for (i = 0; i < argc; i++)
    {
        printf("%s ", argv[i]); /* Printing the string */
    }
    printf("\n");
}
```

Program Explanation

In this C program, we are using command line arguments to print the program name by using argc and argv[] parameters. Using for loop the entered program name and its argument are printed by initializing the value of ‘i’ variable to zero. The loop will execute till the condition that value ‘i’ variable becomes less than the value of argc parameter. Print the program name entered in the command line argument.

advertisement

Runtime Test Cases

```
$ cc arg9.c
$ a.out this is c class by sanfoundry
a.out this is c class by sanfoundry
```

235.C Program to Swap two Numbers using Bitwise Operators

[« Prev](#)

[Next »](#)

This C Program Swaps two Numbers using Bitwise operators.

Here is source code of the C Program to Swap two Numbers using Bitwise operators. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Swap two Numbers using Bitwise operators
3.  */
4. #include <stdio.h>
5. #include <string.h>
6.
7. /* Function Prototype */
8. void swap(int*, int*);
```

```

10.void main()
11.{
12.    int num1, num2;
13.    printf("\nEnter two numbers:");
14.    scanf("%d %d", &num1, &num2);
15.    printf("\nThe numbers before swapping are Number1= %d Number2 = %d", num1, num2);
16.    swap(&num1, &num2); /* Call by Reference to function swap */
17.    printf("\nThe numbers after swapping are Number1= %d Number2 = %d", num1, num2);
18.}
19.
20./* Code to swap two numbers using bitwise operator */
21.void swap(int *x, int *y)
22.{
23.    *x = *x ^ *y;
24.    *y = *x ^ *y;
25.    *x = *x ^ *y;
26.}

```

advertisement

```

$ cc bit27.c
$ a.out

```

```

Enter two numbers:45 76
The numbers before swapping are Number1= 45 Number2=76
The numbers after swapping are Number1= 76 Number2=45

```

236.C Program to Compute First N Fibonacci Numbers using Command Line Arguments

[« Prev](#)

[Next »](#)

This C Program computes first N fibonacci numbers using command line arguments.

Here is source code of the C Program to compute first N fibonacci numbers using command line arguments. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2.  * C Program to Compute First N Fibonacci Numbers using Command Line Arguments
3.  */
4. #include <stdio.h>
5.
6. /* Global Variable Declaration */
7. int first = 0;
8. int second = 1;
9. int third;
10./* Function Prototype */
11.void rec_fibonacci(int);

```

```

12.
13.void main(int argc, char *argv[]){/* Command line Arguments*/
14.{ 
15.    int number = atoi(argv[1]);
16.    printf("%d\t%d", first, second); /* To print first and second number of fibonacci series */
17.    rec_fibonacci(number);
18.    printf("\n");
19.}
20.
21./* Code to print fibonacci series using recursive function */
22.void rec_fibonacci(int num)
23.{ 
24.    if (num == 2) /* To exit the function as the first two numbers are already printed */
25.    {
26.        return;
27.    }
28.    third = first + second;
29.    printf("\t%d", third);
30.    first = second;
31.    second = third;
32.    num--;
33.    rec_fibonacci(num);
34.}

```

advertisement

```

$ cc arg6.c
$ a.out 10
0    1    1    2    3    5    8    13   21   34

```

237.C Program to Create Employee File Name Record that is taken from the Command-Line Argument

[« Prev](#)

[Next »](#)

This C Program creates employee file name record that is taken from the command line argument.

Here is source code of the C Program to create employee file name record that is taken from the command line argument. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2.  * C Program to Create Employee File Name Record that is taken from the Command-Line Argument
3. */
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <errno.h>
7. #include <string.h>
8.
9. struct emprec

```

```

10.{  

11.    int empid;  

12.    char *name;  

13.};  

14.typedef struct emprec emp;  

15.  

16.void insert(char *a);  

17.void display(char *a);  

18.void update(char *a);  

19.int count;  

20.void main(int argc, char *argv[])  

21.{  

22.    int choice;  

23.  

24.    while (1)  

25.    {  

26.        printf("Enter the choice\n");  

27.        printf("1-Insert a new record into file\n2-Display the records\n");  

28.        printf("3-Update the record\n4-Exit\n");  

29.        scanf("%d", &choice);  

30.        switch (choice)  

31.        {  

32.            case 1:  

33.                insert(argv[1]);  

34.                break;  

35.            case 2:  

36.                display(argv[1]);  

37.                break;  

38.            case 3:  

39.                update(argv[1]);  

40.                break;  

41.            case 4:  

42.                exit(0);  

43.            default:  

44.                printf("Enter the correct choice\n");  

45.        }  

46.    }  

47.}  

48.  

49.//* To insert a new record into the file */  

50.void insert(char *a)  

51.{  

52.    FILE *fp1;  

53.    emp *temp1 = (emp *)malloc(sizeof(emp));  

54.    temp1->name = (char *)malloc(200 * sizeof(char)); //allocating memory for pointer  

55.  

56.    fp1 = fopen(a, "a+");  

57.    if (fp1 == NULL)  

58.        perror("");  

59.    else  

60.    {  

61.        printf("Enter the employee id\n");  

62.        scanf("%d", &temp1->empid);  

63.        fwrite(&temp1->empid, sizeof(int), 1, fp1);  

64.        printf("Enter the employee name\n");  

65.        scanf(" %[^\n]", temp1->name);  

66.        fwrite(temp1->name, 200, 1, fp1);  

67.        count++;  

68.    }  

69.    fclose(fp1);  

70.    free(temp1);  

71.    free(temp1->name);  

72.}  

73.  

74.//* To display the records in the file */  

75.void display(char *a)

```

```

76.{
77. FILE *fp1;
78. char ch;
79. int var = count;
80. emp *temp = (emp *)malloc(sizeof(emp));
81. temp->name = (char *)malloc(200*sizeof(char));
82.
83. fp1 = fopen(a, "r");
84. if (count == 0)
85. {
86.     printf("no records to display\n");
87.     return;
88. }
89. if (fp1 == NULL)
90.     perror("");
91. else
92. {
93.     while(var) // display the employee records
94.     {
95.         fread(&temp->empid, sizeof(int), 1, fp1);
96.         printf("%d", temp->empid);
97.         fread(temp->name, 200, 1, fp1);
98.         printf(" %s\n", temp->name);
99.         var--;
100.    }
101. }
102. fclose(fp1);
103. free(temp);
104. free(temp->name);
105.}
106.
107./* To Update the given record */
108.void update(char *a)
109.{ 
110. FILE *fp1;
111. char ch, name[200];
112. int var = count, id, c;
113. emp *temp = (emp *)malloc(sizeof(emp));
114. temp->name = (char *)malloc(200*sizeof(char));
115.
116. fp1 = fopen(a, "r+");
117. if (fp1 == NULL)
118.     perror("");
119. else
120. {
121.     while (var) //displaying employee records so that user enter correct employee id
122.     {
123.         fread(&temp->empid, sizeof(int), 1, fp1);
124.         printf("%d", temp->empid);
125.         fread(temp->name, 200, 1, fp1);
126.         printf(" %s\n", temp->name);
127.         var--;
128.    }
129.     printf("enter which employee id to be updated\n");
130.     scanf("%d", &id);
131.     fseek(fp1, 0, 0);
132.     var = count;
133.     while(var) //loop to update the name of entered employeedid
134.     {
135.         fread(&temp->empid, sizeof(int), 1, fp1);
136.         if (id == temp->empid)
137.         {
138.             printf("enter employee name for update:");
139.             scanf(" %[^\n]s", name);
140.             c = fwrite(name, 200, 1, fp1);
141.             break;

```

```
142.     }
143.     fread(temp->name, 200, 1, fp1);
144.     var--;
145.   }
146.   if (c == 1)
147.     printf("update of the record successfully\n");
148.   else
149.     printf("update unsuccessful enter correct id\n");
150.   fclose(fp1);
151.   free(temp);
152.   free(temp->name);
153. }
154. }
```

advertisement

```
$ cc file2.c
$ a.out emp
Enter the choice
1-Insert a new record into file
2-Display the records
3-Update the record
4-Exit
1
Enter the employee id
100
Enter the employee name
AAA
Enter the choice
1-Insert a new record into file
2-Display the records
3-Update the record
4-Exit
1
Enter the employee id
200
Enter the employee name
BBB
Enter the choice
1-Insert a new record into file
2-Display the records
3-Update the record
4-Exit
1
Enter the employee id
300
Enter the employee name
CCC
Enter the choice
1-Insert a new record into file
2-Display the records
3-Update the record
4-Exit
1
Enter the employee id
400
Enter the employee name
DDD
Enter the choice
1-Insert a new record into file
2-Display the records
3-Update the record
4-Exit
1
Enter the employee id
500
Enter the employee name
```

EEE
Enter the choice
1-Insert a new record into **file**
2-Display the records
3-Update the record
4-Exit
2
100 AAA
200 BBB
300 CCC
400 DDD
500 EEE
Enter the choice
1-Insert a new record into **file**
2-Display the records
3-Update the record
4-Exit
3
100 AAA
200 BBB
300 CCC
400 DDD
500 EEE
enter **which** employee **id** to be updated
200
enter employee name **for** update:CBF
update of the record successfully
Enter the choice
1-Insert a new record into **file**
2-Display the records
3-Update the record
4-Exit
2
100 AAA
200 CBF
300 CCC
400 DDD
500 EEE
Enter the choice
1-Insert a new record into **file**
2-Display the records
3-Update the record
4-Exit
4

238.C Program to Reverse the String using Both Recursion and Iteration

[« Prev](#)

[Next »](#)

This C Program reverse the string using both recursion and iteration.

Here is source code of the C Program to reverse the string using both recursion and iteration. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2.  * C Program to Reverse the String using Both Recursion and Iteration
3.  */
4. #include <stdio.h>
5. #include <string.h>
6.
7. /* Function Prototype */
8. void disp_str1_rec(char *);
9.
10.void main()
11.{  
12.    char str1[100], *ptr;  
13.    int len1 = 0, i;
```

```

14. char ch;
15. printf("Enter the string:\n");
16. scanf("%[^\\n]s", str1);
17. ptr = str1;
18. len1 = strlen(str1);
19. printf("Using iteration:\n");
20. for (i = len1 - 1; i >= 0; i--) /* Iterative loop */
21. {
22.
23.     ch = str1[i];
24.     printf("%c", ch);
25. }
26. printf("Using recursion:\n");
27. disp_str1_rec(ptr);
28.
29.
30./* Code to reverse the string using Recursion */
31.void disp_str1_rec(char *stng)
32.
33.{
34.    char ch;
35.    if (*stng != '\0')
36.    {
37.        ch = *stng;
38.        stng++;
39.        disp_str1_rec(stng);
40.        printf("%c", ch);
41.    }
42.    else
43.    return;
43.

```

advertisement

```

$ cc string21.c
$ a.out
Enter the string:
welcome to sanfoundry's c programming class

```

```

Using iteration:
ssalc gnimmargorp c s'yrduofnas ot emoclew
Using recursion:
ssalc gnimmargorp c s'yrduofnas ot emoclewi

```

239.C Program to Display Every Possible Combination of Two Words from the given 2 String without Displaying Repeated Combinations

[« Prev](#)

[Next »](#)

This C Program Displays Every Possible Combination of Two Words from the given 2 String without Displaying Repeated Combinations.

Here is source code of the C Program to Display Every Possible Combination of Two Words from the given 2 String without Displaying Repeated Combinations. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

1. /*
2.  * C Program to Display Every Possible Combination of Two Words
3.  * from the given 2 String without Displaying Repeated Combinations
4.  */
5. #include <stdio.h>
6. #include <string.h>

```

```

7.
8. void main()
9. {
10.    char str1[50], str2[50], str3[100][100], str4[100][100];
11.    char str5[200][200], temp[200], str[200][200];
12.    int i, j = 0, k = 0, l = 0, m = 0, index = 0, n = 0;
13.    printf("Enter first string\n");
14.    scanf("%[^\\n]s", str1);
15.    printf("Enter second string\n");
16.    scanf(" %[^\\n]s", str2);
17.
18./* code to convert string in 2-D array */
19.    for (i = 0;str1[i] != '\0';i++)
20.    {
21.        if ((str1[i] == ' '))
22.        {
23.            str3[j][k] = '\0';
24.            j++;
25.            k = 0;
26.        }
27.        else
28.        {
29.            str3[j][k] = str1[i];
30.            k++;
31.        }
32.        str3[j][k] = '\0';
33.    }
34.    k = 0;
35.
36.    for (i = 0;str2[i] != '\0';i++)
37.    {
38.        if ((str2[i] == ' '))
39.        {
40.            str4[l][k] = '\0';
41.            l++;
42.            k = 0;
43.        }
44.        else
45.        {
46.            str4[l][k] = str2[i];
47.            k++;
48.        }
49.        str4[l][k] = '\0';
50.    }
51./* Code to make the first string words combination with second */
52.    for (i = 0;i <= j;i++)
53.    {
54.        for (m = 0;m <= l;m++)
55.        {
56.            strcpy(temp, str3[i]);
57.            strcat(temp, str4[m]);
58.            strcpy(str5[index], temp);
59.            index++;
60.        }
61.    }
62.
63./* Code to make the second string words combination with first */
64.    for (i = 0;i <= l;i++)
65.    {
66.        for (m = 0;m <= j;m++)
67.        {
68.            strcpy(temp, str4[m]);
69.            strcat(temp, str3[i]);
70.            strcpy(str5[index], temp);
71.            index++;
72.        }

```

```

73. }
74.
75./* Code to remove the repetitions */
76. for (i = 0;i <= index;i++)
77. {
78.     for (j = i + 1;j <= index;j++)
79.     {
80.         if ((strcmp(str5[i], str5[j]) == 0)
81.         {
82.             for (k = j;k <= index;k++)
83.             {
84.                 strcpy(str5[k], str5[k + 1]);
85.             }
86.             index--;
87.         }
88.     }
89. }
90. for (i = 0;i <= index;i++)
91. {
92.     printf("%c\n", str5[i]);
93. }
94.

```

advertisement

```

$ cc program27.c
$ a.out
Enter first string
welcome to sanfoundry's class
Enter second string
welcome to c programming class
welcomewelcome
welcometo
welcomec
welcomeprogramming
welcomeclass
towelcome
toto
toc
toprogramming
toclass
sanfoundry'swelcome
sanfoundry'sto
sanfoundry'sc
sanfoundry'sprogramming
sanfoundry'sclass
classwelcome
classto
classc
classprogramming
classclass
cwelcome
programmingwelcome
cto
programmingto
welcomesanfoundry's
tosanfoundry's
csanfoundry's
programmingsanfoundry's
cclass
programmingclass

```

240.C Program to Count the Occurrences of each C Keyword using Array Structure

« [Prev](#)

[Next](#) »

This C Program counts the occurrences of each C keyword using array structure. C keywords are the words which have a definition and cannot be used as an identifier.

Here is a source code of the C program to count the occurrences of each C keyword using array structure. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Count the Occurrences of each C Keyword
3. * using Array Structure
4. */
5. #include <stdio.h>
6. #include <string.h>
7. #include <ctype.h>
8. #define KEYMAX 32
9.
10. struct keyword
```

```

11.
12. char word[10];
13. int occur;
14.};
15.
16.int binarysearch(char [], struct keyword[]);
17.
18.int main()
19.{ 
20. int i = 0, j = 0, pos;
21. char string[100], unit[20], c;
22. struct keyword key[32] = {"auto", 0, "break", 0, "case", 0,
23.           "char", 0, "const", 0, "continue", 0,
24.           "default", 0, "do", 0, "double", 0,
25.           "else", 0, "enum", 0, "extern", 0,
26.           "float", 0, "for", 0, "goto", 0,
27.           "if", 0, "int", 0, "long", 0,
28.           "register", 0, "return", 0, "short", 0,
29.           "signed", 0, "sizeof", 0, "static", 0,
30.           "struct", 0, "switch", 0, "typedef", 0,
31.           "union", 0, "unsigned", 0, "void", 0,
32.           "volatile", 0, "while", 0,};
33.
34. printf("Enter string: ");
35. do
36. {
37.   fflush(stdin);
38.   c = getchar();
39.   string[i++] = c;
40.
41. } while (c != '\n');
42. string[i - 1] = '\0';
43. printf("The string entered is: %s\n", string);
44. for (i = 0; i < strlen(string); i++)
45. {
46.   while (i < strlen(string) && string[i] != ' ' && isalpha(string[i]))
47.   {
48.     unit[j++] = tolower(string[i++]);
49.   }
50.   if (j != 0)
51.   {
52.     unit[j] = '\0';
53.     pos = binarysearch(unit, key);
54.     j = 0;
55.     if (pos != -1)
56.     {
57.       key[pos].occur++;
58.     }
59.   }
60. }
61. printf("*****\n Keyword\tCount\n*****\n");
62. for (i = 0; i < KEYMAX; i++)
63. {
64.   if (key[i].occur)
65.   {
66.     printf("  %s\t %d\n", key[i].word, key[i].occur);
67.   }
68. }
69.
70. return 0;
71.}
72.
73.int binarysearch(char *word, struct keyword key[])
74.{ 
75.   int low, high, mid;
76.

```

```

77. low = 0;
78. high = KEYMAX - 1;
79. while (low <= high)
80. {
81.     mid = (low + high) / 2;
82.     if (strcmp(word, key[mid].word) < 0)
83.     {
84.         high = mid - 1;
85.     }
86.     else if (strcmp(word, key[mid].word) > 0)
87.     {
88.         low = mid + 1;
89.     }
90.     else
91.     {
92.         return mid;
93.     }
94. }
95.
96. return -1;
97.

```

advertisement

```

$ gcc keywordoccur.c
$ ./a.out
Enter string: break, float and double are c keywords. float and double are primitive data types.
The string entered is: break, float and double are c keywords. float and double are primitive data types.
*****
Keyword Count
*****
break      1
double      2
float       2

```

241.C Program to Accept 2 String & check whether all Characters in first String is Present in second String & Print

[« Prev](#)

[Next »](#)

This is a C Program to accept 2 string & check whether all characters in first string is present in second string & print.

Problem Description

This C Program checks whether all characters used in first string are present in second string.

Problem Solution

Take input from the user and perform string operations as shown in the program below.

advertisement

Program/Source Code

Here is a source code of the C program to check whether all characters used in first string are present in the second string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Accept 2 String & check whether all Characters
 * in first String is Present in second String & Print
 */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#define CHAR_SIZE 26

void alphacheck(char *, int []);
void create(char *, int[]);

int main()
{
    char str1[50], str2[50];
    int a1[CHAR_SIZE] = {0}, a2[CHAR_SIZE] = {0}, i;
    char str1_alpha[CHAR_SIZE], str2_alpha[CHAR_SIZE];

    printf("Enter string1: ");
    scanf("%s", str1);
    printf("Enter string2: ");
    scanf("%s", str2);
    alphacheck(str1, a1);
    alphacheck(str2, a2);
    create(str1_alpha, a1);
    create(str2_alpha, a2);
    if (strcmp(str1_alpha, str2_alpha) == 0)
    {
        printf("All characters match in %s and %s.\n", str1, str2);
        printf("The characters that match are: ");
        for (i = 0; i < strlen(str1_alpha); i++)
        {
            printf("%c, ", str1_alpha[i]);
        }
        printf("\n");
    }
    else
    {
        printf("All characters do not match in %s and %s.\n", str1, str2);
    }
    return 0;
}

void alphacheck(char *str, int a[])
{
    int i, index;

    for (i = 0; i < strlen(str); i++)
    {
        str[i] = tolower(str[i]);
        index = str[i] - 'a';
        if (!a[index])
        {
            a[index] = 1;
        }
    }
}
```

```

void create(char *str, int a[])
{
    int i, j = 0;

    for (i = 0; i < CHAR_SIZE; i++)
    {
        if (a[i])
        {
            str[j++] = i + 'a';
        }
    }
    str[j] = '\0';
}

```

Program Explanation

In this C Program, we are reading two strings using ‘str1’ and ‘str2’ variables. The alphacheck() function is used to accept 2 strings & check whether all characters in first string is present in second string.

advertisement

For loop the tolower() function is used to convert all the characters present in the string to lower case. Compute the difference between each character present in the ‘str[]’ array variable by ‘a’ character value. If condition statement is used to assign the value of ‘a[]’ array variable with base index ‘index’ variable to 1.

The create() function is used to accept 2 strings from str1[] and str2[] array variables respectively. For loop is used to assign the character value to the ‘str[]’ array variable.

If else condition statement is used to check the characters in first string is present in second string by using strcmp() function. If the condition is true then execute the statement, using for loop print the statement as the characters that match in the two strings.

advertisement

Otherwise, if the condition is false, then execute the else statement. Print the statement as all characters do not match in the string1 and string2.

Runtime Test Cases

```

$ cc allchar.c
$ ./a.out
Enter string1: aspired
Enter string2: despair
All characters match in aspired and despair.
The characters that match are: a, d, e, i, p, r, s,

```

242.C Program to Count the Number of all Repeated Words in a String & Display the Word with its Frequency

[« Prev](#)

[Next »](#)

This C Program counts the number of all repeated words in a string and displays the repeated words with it's frequency.

Here is a source code of the C program that counts the number of all repeated words in a string display the word with it's frequency. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Count the Number of all Repeated Words
3. * in a String & Display the Word with its Frequency
4. */
5. #include <stdio.h>
6. #include <string.h>
7. #include <ctype.h>
8.
9. struct detail
10. {
11.     char word[20];
12.     int freq;
13. };
```

```

14.
15.int update(struct detail [], const char [], int);
16.
17.int main()
18.{
19.    struct detail s[10];
20.    char string[100], unit[20], c;
21.    int i = 0, freq = 0, j = 0, count = 0, num = 0;
22.
23.    for (i = 0; i < 10; i++)
24.    {
25.        s[i].freq = 0;
26.    }
27.    printf("Enter string: ");
28.    i = 0;
29.    do
30.    {
31.        fflush(stdin);
32.        c = getchar();
33.        string[i++] = c;
34.
35.    } while (c != '\n');
36.    string[i - 1] = '\0';
37.    printf("The string entered is: %s\n", string);
38.    for (i = 0; i < strlen(string); i++)
39.    {
40.        while (i < strlen(string) && string[i] != ' ' && isalnum(string[i]))
41.        {
42.            unit[j++] = string[i++];
43.        }
44.        if (j != 0)
45.        {
46.            unit[j] = '\0';
47.            count = update(s, unit, count);
48.            j = 0;
49.        }
50.    }
51.
52.    printf("*****\nWord\tFrequency\n*****\n");
53.    for (i = 0; i < count; i++)
54.    {
55.        printf("%s\t %d\n", s[i].word, s[i].freq);
56.        if (s[i].freq > 1)
57.        {
58.            num++;
59.        }
60.    }
61.    printf("The number of repeated words are %d.\n", num);
62.
63.    return 0;
64.}
65.
66.int update(struct detail s[], const char unit[], int count)
67.{
68.    int i;
69.
70.    for (i = 0; i < count; i++)
71.    {
72.        if (strcmp(s[i].word, unit) == 0)
73.        {
74.            s[i].freq++;
75.
76.            return count;
77.        }
78.    }
79.    /*If control reaches here, it means no match found in struct*/

```

```
80. strcpy(s[count].word, unit);
81. s[count].freq++;
82.
83. /*count represents the number of fields updated in array s*/
84. return (count + 1);
85.}
```

advertisement

```
$ cc wordfrequency.c
$ ./a.out
Enter string: hello world hello program hello C
The string entered is: hello world hello program hello C
*****
Word      Frequency
*****
hello      3
world      1
program    1
C          1
The number of repeated words are 1.
```

243.C Program to Find the Most/Least Repeated Character in the String

[« Prev](#)

[Next »](#)

This is a C program to find the most/least repeated character in the string.

Problem Description

This C Program finds the most/least repeated character in the string.

Problem Solution

Take input from the user and displays most/least repeated characters as shown in the program below.

advertisement

Program/Source Code

Here is a source code of the C program to find the most/least repeated character in the string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```

/*
*C Program to Find the Most/Least Repeated Character in the String
*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>

struct detail
{
    char c;
    int freq;
};

int main()
{
    struct detail s[26];
    char string[100], c;
    int max[26] = {0}, min[26] = {0};
    int i = 0, index, maxcount = 1, mincount = 1000, j;

    for (i = 0; i < 26; i++)
    {
        s[i].c = i + 'a';
        s[i].freq = 0;
    }
    printf("Enter string: ");
    i = 0;
    do
    {
        fflush(stdin);
        c = getchar();
        string[i++] = c;
        if (c == '\n')
        {
            break;
        }
        else if (!isalpha(c))
        {
            continue;
        }
        c = tolower(c);
        index = c - 'a';
        s[index].freq++;
    } while (1);
    string[i - 1] = '\0';
    printf("The string entered is: %s\n", string);
    for (i = 0; i < 26; i++)
    {
        if (s[i].freq)
        {
            if (maxcount < s[i].freq)
            {
                for (j = 0; j < 26; j++)
                {
                    max[j] = 0;
                }
                max[i] = 1;
                maxcount = s[i].freq;
            }
            else if (maxcount == s[i].freq)
            {
                max[i] = 1;
            }
            if (mincount >= s[i].freq)
            {
                if (mincount == s[i].freq)

```

```

    {
        min[i] = 1;
    }
    else
    {
        for (j = 0; j < 26; j++)
        {
            min[j] = 0;
        }
        min[i] = 1;
        mincount = s[i].freq;
    }
}
printf("The most repeated characters are: ");
for (i = 0; i < 26; i++)
{
    if (max[i])
    {
        printf("%c ", i + 'a');
    }
}
printf("\nThe least repeated characters are: ");
for (i = 0; i < 26; i++)
{
    if (min[i])
    {
        printf("%c ", i + 'a');
    }
}
printf("\n");

return 0;
}

```

Program Explanation

In this C Program, do while loop is used for reading a string using ‘c’ variable. If else condition statement is used to check that the entered character is empty i.e, is space or enter. If the condition is true then execute the statement.

advertisement

Otherwise, if the condition is false, then execute the else if condition statement. Check the entered character is numeric character or not. If the condition is true, then execute the statement. Convert the value of string variable into lower case.

For loop is used to find the most or least repeated character in the string. Nested if else condition statement is used to check the value of ‘maxcount’ variable is less than the number of characters entered. If the condition is true, then execute the statement.

Otherwise if the condition is false, then execute the else if condition statement. Check the value of ‘maxcount’ variable is equal to the number of characters entered. If the condition is true, then execute the statement and initialize the value of ‘max[]’ variable as 1.

advertisement

Another nested if condition statement is used to check the value of ‘mincount’ variable is greater than or equal to the number of characters entered. If the condition is true, then execute the statement. Again check the condition that the value of ‘mincount’ variable is equal to the number of characters occurred in the string.

If the condition is true then execute the statement and assign the value of 1 to the ‘min[]’ array variable. Otherwise, if the condition is false then execute the else condition statement. Using for loop prints the most and least characters in the string.

Runtime Test Cases

```
$ gcc minmaxchar.c
$ ./a.out
Enter string: I love C programming
The string entered is: I love C programming
The most repeated characters are: g i m o r
The least repeated characters are: a c e l n p v
```

244.C Program to Input a String with atleast one Number, Print the Square of all the Numbers in a String

[« Prev](#)

[Next »](#)

This is a C Program to input a string with atleast one number, print the square of all the numbers in a string.

Problem Description

This C Program takes an input string with atleast one number and prints the square of all the numbers in the string.

Problem Solution

Take input from the user and perform string operations as shown in the program below.

advertisement

Program/Source Code

Here is a source code of the C program to print the squares of all the numbers entered in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Input a String with atleast one Number, Print
 * the Square of all the Numbers in a String
 */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

struct detail
{
    int number;
    int square;
};

int update(struct detail [], int, int);
int toint(char []);

int main()
{
    struct detail s[10];
    char unit[20], string[100];
    char c;
    int num, i, j = 0, count = 0;

    printf("Enter string: ");
    i = 0;
    do
    {
        fflush(stdin);
        c = getchar();
        string[i++] = c;

    } while (c != '\n');
    string[i - 1] = '\0';
    printf("The string entered is: %s\n", string);
    for (i = 0; i < strlen(string); i++)
    {
        while (i < strlen(string) && !isspace(string[i]))
        {
            unit[j++] = string[i++];
        }
        if (j != 0)
        {
            unit[j] = '\0';
            num = toint(unit);
            count = update(s, num, count);
            j = 0;
        }
    }
    printf("*****\nNumber\tSquare\n*****\n");
    for (i = 0; i < count; i++)
    {
        printf("%d\t %d\n", s[i].number, s[i].square);
    }

    return 0;
}

int update(struct detail s[], int num, int count)
{
    s[count].number = num;
    s[count].square = num * num;
```

```

    return (count + 1);
}

int toint(char str[])
{
    int len = strlen(str);
    int i, num = 0;

    for (i = 0; i < len; i++)
    {
        num = num + ((str[len - (i + 1)] - '0') * pow(10, i));
    }

    return num;
}

```

Program Explanation

This C program, we are reading the string using ‘string[]’ array variable. Do while loop is used to accept the input value of the string using getchar() function and assign the value to ‘c’ character variable.

advertisement

While loop is used to check the value of character variable ‘c’ is not equal to empty character null. If the condition is true, then execute the iteration of the loop. For loop is used to compute the square of all the numbers.

While loop is used to check the value of ‘i’ variable is less than the length of the string and string should not consist any empty space using isspace() function using logical AND operator. If the condition is true then assign the value of ‘string[]’ array variable to ‘unit[]’ variable.

If condition statement is used to check the value of ‘j’ variable is not equal to 0. If the condition is true, then execute the statement. Assign the value of ‘unit[]’ array variable as null. The toint() function is used to compute the square of all the numbers in a string.

advertisement

For loop is used to compute the square of all the numbers in a string. Initialize the value of ‘i’ variable to 0 and check the value of ‘i’ variable is less than the value of ‘len’ variable. If the condition is true, then execute the loop.

Compute the power value of 10 to the power of value of ‘i’ variable. Multiply the length of the string with the power value. Add the resulted value with the ‘num’ variable and return the value.

In update() function compute the square of the num variable value and assigning to the structure variable s[count].square’. Print the square of all the numbers in a string using printf statement.

advertisement

Runtime Test Cases

```

$ gcc numbersquare.c -lm
$ ./a.out
Enter string: 1 2 3 4 5
The string entered is: 1 2 3 4 5
*****
Number   Square
*****
1       1
2       4

```

3	9
4	16
5	25

245.C Program to Find the Largest & Smallest possible Word which is a Palindrome

« [Prev](#)

[Next](#) »

This C Program finds the largest and smallest possible word which is a palindrome. A palindrome is a word that reads the same backward as forward, e.g., madam.

Here is a source code of the C program to find the largest and smallest possible word which is a palindrome. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. /*
2. * C Program to Find the Largest & Smallest possible
3. * Word which is a Palindrome
4. */
5. #include <stdio.h>
6. #include <string.h>
7. #include <ctype.h>
8.
9. int palin_check(char []);
10.
11.int main()
12.{  
13. char string[100], word[20], max[20], min[20], c;  
14. int i = 0, j = 0, flag = 0, check;  
15.
16. printf("Enter string: ");  
17. i = 0;  
18. do  
19. {  
20. fflush(stdin);
```

```

21.     c = getchar();
22.     string[i++] = c;
23.
24. } while (c != '\n');
25. string[i - 1] = '\0';
26. for (i = 0; i < strlen(string); i++)
27. {
28.     while (i < strlen(string) && !isspace(string[i]) && isalnum(string[i]))
29.     {
30.         word[j++] = string[i++];
31.     }
32.     if (j != 0)
33.     {
34.         word[j] = '\0';
35.         check = palin_check(word);
36.         if (check)
37.         {
38.             if (!flag)
39.             {
40.                 flag = !flag;
41.                 strcpy(max, word);
42.                 strcpy(min, word);
43.             }
44.             if (strlen(word) > strlen(max))
45.             {
46.                 strcpy(max, word);
47.             }
48.             if (strlen(word) < strlen(min))
49.             {
50.                 strcpy(min, word);
51.             }
52.         }
53.         j = 0;
54.     }
55. }
56. if (flag)
57. {
58.     printf("The largest palindrome is '%s' and smallest palindrome is '%s' in '%s'.\n", max, min, string);
59. }
60. else
61. {
62.     printf("No palindrome words exists in '%s'.\n", string);
63. }
64.
65. return 0;
66.}
67.
68.int palin_check(char str[])
69.
70. int i, len;
71.
72. len = strlen(str);
73. for (i = 0; i < len / 2; i++)
74. {
75.     if (tolower(str[i]) != tolower(str[len - (i + 1)]))
76.     {
77.         return 0;
78.     }
79. }
80.
81. return 1;
82.

```

advertisement

```
$ gcc largesmallpalin.c
$ ./a.out
```

```
Enter string: hello madam we speak malayalam
The largest palindrome is 'malayalam' and smallest palindrome is 'madam' in 'hello madam we speak mal
```

246.C Program to Count the Number of Repeated Occurrences of a particular Word in a String

[« Prev](#)

[Next »](#)

This is a C Program to count the number of repeated occurrences of a particular word in a string.

Problem Description

This C Program counts the number of repeated occurrences of a particular word in a string.

Problem Solution

Take input from the user and performs string operations to count the repeated occurrences of a particular word as shown in the program below.

advertisement

Program/Source Code

Here is a source code of the C program that counts the number of repeated occurrences of a particular word in a string. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
/*
 * C Program to Count the Number of Repeated Occurrences
 * of a particular Word in a String
```

```

/*
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main()
{
    char string[100], word[20], unit[20], c;
    int i = 0, j = 0, count = 0;

    printf("Enter string: ");
    i = 0;
    do
    {
        fflush(stdin);
        c = getchar();
        string[i++] = c;

    } while (c != '\n');
    string[i - 1] = '\0';
    printf("Enter the word you want to find: ");
    scanf("%s", word);
    for (i = 0; i < strlen(string); i++)
    {
        while (i < strlen(string) && !isspace(string[i]) && isalnum(string[i]))
        {
            unit[j++] = string[i++];
        }
        if (j != 0)
        {
            unit[j] = '\0';
            if (strcmp(unit, word) == 0)
            {
                count++;
            }
            j = 0;
        }
    }

    printf("The number of times the word '%s' found in '%s' is '%d'.\n", word, string, count);
}

```

Program Explanation

In this C Program, we are reading the word to find using the ‘word’ variable. For loop is used to count the number of repeated occurrences of a particular word in a string.

advertisement

While loop is used to check that the length, space, and alphanumeric number string is less than the occurring string. If the condition is true, then execute the iteration of the loop. If condition statement is used to check that the value of ‘j’ variable is not equal to 0.

If the condition is true then execute the statement, if statement is used to check that the string compared is equal to 0. If the condition is true then execute the statement. Print number of repeated occurrences of a particular word in a string.

Runtime Test Cases

```

$ cc givenword.c
$ ./a.out
Enter string: hello world hello program hello C
Enter the word you want to find: hello

```

The number of times the word 'hello' found in 'hello world hello program hello C' is 3.

247.C Program to Implement Bit Array

[« Prev](#)

[Next »](#)

This is a C Program to implement Bit Array. A bit array is an array data structure that compactly stores bits. It can be used to implement a simple set data structure. A bit array is effective at exploiting bit-level parallelism in hardware to perform operations quickly.

A typical bit array stores b_t bits, where t is the number of bits in the unit of storage, such as a byte or word, and b is some non-negative integer. If t does not divide the number of bits to be stored, some space is wasted due to internal fragmentation.

Here is source code of the C Program to Implement Bit Array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

advertisement

```
1. #include <stdio.h>
2.
3. #define SIZE (58) /* amount of bits */
4. #define ARRAY_SIZE(x) (x/8+(!(x%8)))
5.
6. char get_bit(char *array, int index);
7. void toggle_bit(char *array, int index);
8. void toggle_bit(char *array, int index) {
9.     array[index / 8] ^= 1 << (index % 8);
10.}
11.
12. char get_bit(char *array, int index) {
13.     return 1 & (array[index / 8] >> (index % 8));
14.}
15. int main(void) {
16.     /* initialize empty array with the right size */
```

```
17. char x[ARRAY_SIZE(SIZE)] = { 0 };
18. int i;
19.
20. for (i = 0; i < SIZE; i += 2)
21.     toggle_bit(x, i);
22. toggle_bit(x, 56);
23. for (i = 0; i < SIZE; i++)
24.     printf("%d: %d\n", i, get_bit(x, i));
25.
26. return 0;
27.}
```

Output:

```
$ gcc BitArray.c
$ ./a.out
```

```
0: 1
1: 0
2: 1
3: 0
4: 1
5: 0
6: 1
7: 0
8: 1
9: 0
10: 1
11: 0
12: 1
13: 0
14: 1
15: 0
16: 1
17: 0
18: 1
19: 0
20: 1
21: 0
22: 1
23: 0
24: 1
25: 0
26: 1
27: 0
28: 1
29: 0
30: 1
31: 0
32: 1
33: 0
34: 1
35: 0
36: 1
37: 0
38: 1
39: 0
40: 1
41: 0
42: 1
43: 0
44: 1
45: 0
46: 1
47: 0
48: 1
```

49: 0
50: 1
51: 0
52: 1
53: 0
54: 1
55: 0
56: 0
57: 0

248.C Program to Implement Variable Length Array

« [Prev](#)

[Next](#) »

This is a C Program to implement variable length array(Vectors). An array (vector) is a common-place data type, used to hold and describe a collection of elements. These elements can be fetched at runtime by one or more indices (identifying keys). A distinguishing feature of an array compared to a list is that they allow for constant-time random access lookup, compared to the latters sequential access. Resizable arrays allow for an unspecified upper-bound of collection elements at runtime, and are conceptually similar to a list. These dynamic arrays are more complicated and less used in introduction to its compatriot list, which is dynamic by nature. Using C as the language of implementation this post will guide you through building a simple vector data-structure. The structure will take advantage of a fixed-size array, with a counter invariant that keeps track of how many elements are currently present. If the underlying array becomes exhausted, the addition operation will re-allocate the contents to a larger size, by way of a copy.

Here is source code of the C Program to Implement Variable Length Array. The C program is successfully compiled and run on a Linux system. The program output is also shown below.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. #ifndef VECTOR_H
5. #define VECTOR_H
6.
7. #define VECTOR_INIT_CAPACITY 4
8.
9. #define VECTOR_INIT(vec) vector vec; vector_init(&vec)
10.#define VECTOR_ADD(vec, item) vector_add(&vec, (void *) item)
11.#define VECTOR_SET(vec, id, item) vector_set(&vec, id, (void *) item)
12.#define VECTOR_GET(vec, type, id) (type) vector_get(&vec, id)
```

```
13.#define VECTOR_DELETE(vec, id) vector_delete(&vec, id)
14.#define VECTOR_TOTAL(vec) vector_total(&vec)
15.#define VECTOR_FREE(vec) vector_free(&vec)
16.
17.typedef struct vector {
18.    void **items;
19.    int capacity;
20.    int total;
21.} vector;
22.
23.void vector_init(vector *);
24.int vector_total(vector *);
25.static void vector_resize(vector *, int);
26.void vector_add(vector *, void *);
27.void vector_set(vector *, int, void *);
28 void *vector_get(vector *, int);
29.void vector_delete(vector *, int);
30.void vector_free(vector *);
31.
32.#endif
33.
34.void vector_init(vector *v) {
35.    v->capacity = VECTOR_INIT_CAPACITY;
36.    v->total = 0;
37.    v->items = malloc(sizeof(void *) * v->capacity);
38.}
39.
40.int vector_total(vector *v) {
41.    return v->total;
42.}
43.
44.static void vector_resize(vector *v, int capacity) {
45.#ifdef DEBUG_ON
46.    printf("vector_resize: %d to %d\n", v->capacity, capacity);
47.#endif
48.
49.    void **items = realloc(v->items, sizeof(void *) * capacity);
50.    if (items) {
51.        v->items = items;
52.        v->capacity = capacity;
53.    }
54.}
55.
56.void vector_add(vector *v, void *item) {
57.    if (v->capacity == v->total)
58.        vector_resize(v, v->capacity * 2);
59.    v->items[v->total++] = item;
60.}
61.
62.void vector_set(vector *v, int index, void *item) {
63.    if (index >= 0 && index < v->total)
64.        v->items[index] = item;
65.}
66.
67.void *vector_get(vector *v, int index) {
68.    if (index >= 0 && index < v->total)
69.        return v->items[index];
70.    return NULL;
71.}
72.
73.void vector_delete(vector *v, int index) {
74.    if (index < 0 || index >= v->total)
75.        return;
76.
77.    v->items[index] = NULL;
78.    int i;
```

```

79. for (i = 0; i < v->total - 1; i++) {
80.     v->items[i] = v->items[i + 1];
81.     v->items[i + 1] = NULL;
82. }
83.
84. v->total--;
85.
86. if (v->total > 0 && v->total == v->capacity / 4)
87.     vector_resize(v, v->capacity / 2);
88. }
89.
90. void vector_free(vector *v) {
91.     free(v->items);
92. }
93.
94. int main(void) {
95.     int i;
96.
97.     vector v;
98.     vector_init(&v);
99.
100.    vector_add(&v, "Bonjour");
101.    vector_add(&v, "tout");
102.    vector_add(&v, "le");
103.    vector_add(&v, "monde");
104.
105.    for (i = 0; i < vector_total(&v); i++)
106.        printf("%os ", (char *) vector_get(&v, i));
107.    printf("\n");
108.
109.    vector_delete(&v, 3);
110.    vector_delete(&v, 2);
111.    vector_delete(&v, 1);
112.
113.    vector_set(&v, 0, "Hello");
114.    vector_add(&v, "World");
115.
116.    for (i = 0; i < vector_total(&v); i++)
117.        printf("%os ", (char *) vector_get(&v, i));
118.    printf("\n");
119.
120.    vector_free(&v);
121. }
```

Output:

advertisement

```
$ gcc VariableLengthArray.c
$ ./a.out
```

```
Bonjour tout le monde
Hello World
```

249.C Program to Convert a Given Time in 12 AM-PM Format to 24 Hour Military Format

[« Prev](#)

[Next »](#)

This is a C Program to convert 12-hour AM-PM time format to 24-hour Military time format.

Problem Description

Given a time in 12-hour AM/PM format, convert it to military (24-hour) time. User has to take input as a string containing a time in 12-hour clock format (i.e.: hh:mm:ssAM or hh:mm:ssPM), **where $01 \leq \text{hh} \leq 12$ and $00 \leq \text{mm,ss} \leq 59$.**

Expected Input and Output

Input:- 09:15:55PM

Output:- 21:15:55

advertisement

Input:- 12:00:00AM

Output:- 00:00:00

Input:- 03:55:50AM

Output:- 03:55:50

Problem Solution

- Take input as taken above in sample inputs. (" %d:%d:%d%s ", &hh,&mm,&ss,a) or you can provide inputs to 'hh', 'mm', 'ss' and 'a' separately.
- Check and compare if the string 'a' at the end of input is 'AM' or 'PM'.
- Check the value of hh, and solve accordingly.

advertisement

Program/Source Code

Here is source code of the C Program for conversion of 12-hour AM-PM time format to 24-hour military time format. The program is successfully compiled and tested using Codeblocks gnu/gcc compiler on windows 10. The program output is also shown below.

```

1. /* C Program for converting 12-hour time format to 24-hour time format. */
2. #include<stdio.h>
3. #include<string.h>
4. int main()
5. {
6.     int hh, mm, ss;
7.     char a[3];
8.     printf("Enter hours 'hh' \t");
9.     scanf("%d", &hh);
10.    printf("Enter minutes 'mm' \t");
11.    scanf("%d", &mm);
12.    printf("Enter seconds 'ss' \t");
13.    scanf("%d", &ss);
14.    printf("Enter string 'am' or 'pm' \t");
15.    scanf("%s", &a);
16. /*
17. * user is allowed to enter time only in 12-hour format
18. * so that 'hh' cannot be greater than 12.
19. */
20. if(hh <= 12 && mm <= 59 && ss <= 59)
21. {
22.     if((strcmp(a,"PM") == 0) || (strcmp(a,"pm") == 0)
23.         && (hh != 0) && (hh != 12))
24.     {
25.         hh = hh + 12;
26.     }
27.     if((strcmp(a,"AM") == 0) || (strcmp(a,"am") == 0) && (hh == 12))
28.     {
29.         hh = 0;
30.     }
31.     printf("The obtained 24-hour format of input is \t");
32.     printf("%02d:%02d:%02d", hh, mm, ss);
33.     printf("\n\n");
34. }
35. else
36. {
37.     printf("Provide the correct inputs.");
38. }
39. return 0;
40. }
```

Program Explanation

- The user will give input in 12-hour format, which will contain 4 variables, hh for hours, mm for minutes, ss for seconds and a string 'a' for denoting 'AM' or 'PM'.

advertisement

2. After taking the input the user will check if it is 'PM' and value of 'hh' is anything apart from 00 or 12, it'll be directly added by 12 and mm,ss will remain same. For example, if the user inputs hh as 05 'PM', in 24-hour format 05 pm = 17 which is nothing but 05+12.

3. But if it is 'AM' and value of hh is 12, the value of hh will be made = 0 because after 23rd hour it'll start again from 00, and for the remaining cases if it is 'AM' then the time in 12-hour and 24-hour format will remain same. For example, if the user inputs time as 11:47:56AM the output will be 11:47:56 same as it was in 12-hour format, but if the user enters 12:55:21AM the output will be 00:55:21 because the range of hh's value is between 00 and 23 both inclusive and immediately after 23rd hour we will be getting 00th hour not 24th.

Runtim Test Cases

1. Enter hours 'hh' 09
Enter minutes 'mm' 15
Enter seconds 'ss' 55
Enter string 'am' or 'pm' pm
The obtained 24-hour format of input is 21:15:55

2. Enter hours 'hh' 12
Enter minutes 'mm' 00
Enter seconds 'ss' 00
Enter string 'am' or 'pm' am
The obtained 24-hour format of input is 00:00:00

3. Enter hours 'hh' 03
Enter minutes 'mm' 55
Enter seconds 'ss' 50
Enter string 'am' or 'pm' am
The obtained 24-hour format of input is 03:55:50

4. Enter hours 'hh' 23
Enter minutes 'mm' 13
Enter seconds 'ss' 11
Enter string 'am' or 'pm' am
Provide the correct inputs.

250.C Program to find out the Power of Any Number using Loop

« [Prev](#)

This is a C Program to find out the power of any number using loop.

Problem Description

We have to create a C Program which calculates the power of any number, by taking the values of base and exponent from the user.

Expected Input and Output

1. When base and exponent both are positive

advertisement

If the user provides the value of Base as 2 and Exponent as 5,
then the power i.e base^{exponent} will be 32

2. When Exponent = 0

As we know if the Power of any number is 0, its value is = 1.
Here if the user enters Base as 5 and Exponent as 0,
the output will be 1.

3. When Base is Negative

advertisement

If the user enters a negative Base say -2 and Exponent as 3,

we will have -8 as Power in this case.

4. When Exponent is Negative

advertisement

If the user enters a negative Exponent say -2 and Base as 3,
we will have 0.1111 as Power in this case as the equation will become $1/(3^2)$.

Problem Solution

We have to take two inputs from the user, base and exponent. We have to run a loop from 1 to exponent and calculate the power by multiplying power with base in every iteration.

Program/Source Code

Here is source code of the C Program to find the power of a number using a loop. The program is successfully compiled and tested using Codeblocks gnu/GCC compiler on Windows 10. The program output is also shown below.

```
1. /*
2.  * C program to find power of any number using for loop
3.  */
4.
5. #include <stdio.h>
6.
7. int main()
8. {
9.     int base, exponent;
10.    float power = 1;
11.    int i;
12.
13.    /* Take base and exponent as input*/
14.    printf("Enter base: ");
15.    scanf("%d", &base);
16.    printf("Enter exponent: ");
17.    scanf("%d", &exponent);
18.    int expo = exponent;
19.    while (expo < 0)
20.    {
21.        {
22.            power = power/base;
23.            expo++;
24.        }
25.    }
26.
27.    if(exponent >0)
28.    {
29.
30.        /*Calculate power */
31.        for(i = 1; i <= exponent; i++)
32.        {
33.            power = power * base;
34.        }
35.    }
36.    printf("%d ^ %d = %f", base, exponent, power);
37.
38.    return 0;
39.}
```

Program Explanation

1. In order to calculate the power of a number using a loop we have to take two inputs from user, base and the exponent. We have to initialize a variable named power as 1.
2. Power variable stores the final value of the power.

3. Run a loop starting from index value 1 till the value of exponent, and multiply power with base and store the result in power in each iteration. After it comes out of the loop, power has the output value in it, we just need to print that.

advertisement

Runtime Test Cases

1. Enter base: 2

Enter exponent: 5

$2^5 = 32.000000$

2. Enter base: 5

Enter exponent: 0

$5^0 = 1.000000$

3. Enter base: -2

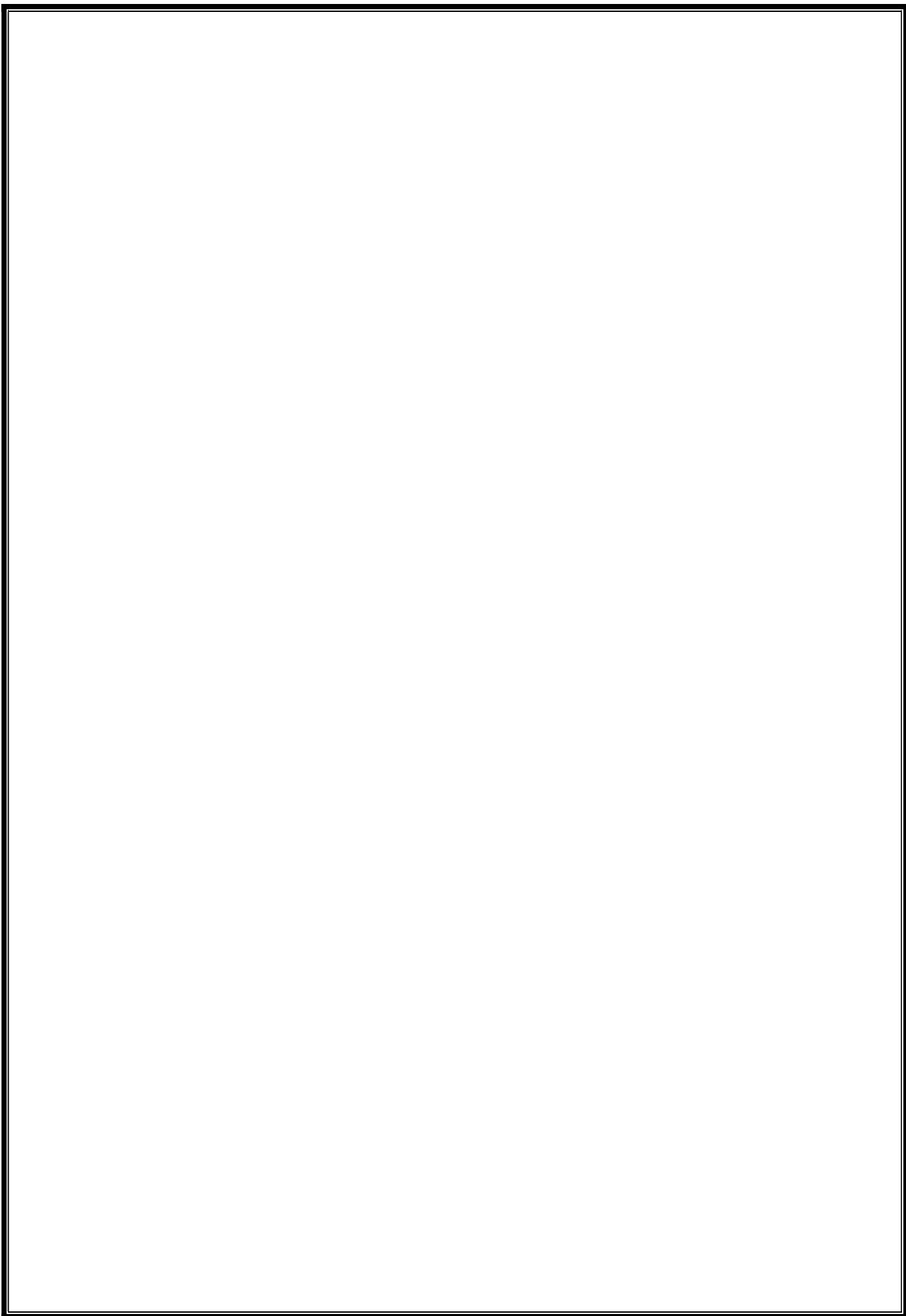
Enter exponent: 3

$-2^3 = -8.000000$

4. Enter base: 3

Enter exponent: -2

$3^{-2} = 0.111111$



JAVA PROGRAMS

1. Java Program to Multiply given Number by 4 using Bitwise Operators

[« Prev](#)

[Next »](#)

This is a Java Program to Multiply given Number by 4 using Bitwise Operators.

Enter the number you wish to multiply as input. After that we use left shift by 2 to multiply given input by four and hence get the output.

Here is the source code of the Java Program to Multiply given Number by 4 using Bitwise Operators. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. import java.util.Scanner;
2. public class Multiply_Bitwise
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number:");
9.         n = s.nextInt();
10.        int mul = n << 2;
11.        System.out.println("Answer:"+mul);
12.    }
13.}
```

Output:

```
$ javac Multiply_Bitwise.java
$ java Multiply_Bitwise
```

```
Enter the number:5
Answer:20
```

2.Java Program to Show the Implementation of Interface

[Next »](#)

This is a Java program to show the Implementation of Interface. We have to enter the length and breadth of the rectangle as an input to get Area of rectangle as output. An interface in Java is similar to a class, but the body of an interface can include only abstract methods and final fields (constants). A class implements an interface by providing code for each method declared by the interface.

Here is the source code of the Java Program to show the Partial Implementation of Interface. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. import java.util.Scanner;
2.
3. interface area
4. {
5.     public void dimensions();
6.     public void area();
7. }
8. public class Interface_Implementation implements area
9. {
10.    int length,breadth,area;
11.    public void dimensions()
12.    {
13.        Scanner s=new Scanner(System.in);
14.        System.out.print("Enter length:");
15.        length=s.nextInt();
16.        System.out.print("Enter breadth:");
17.        breadth=s.nextInt();
18.    }
19.    public void area()
20.    {
21.        area=length*breadth;
22.        System.out.print("Area:"+area);
23.    }
24.    public static void main(String[] args)
25.    {
26.        Interface_Implementation obj=new Interface_Implementation();
27.        obj.dimensions();
28.        obj.area();
29.    }
30.}
```

Output:

advertisement

```
$ javac Interface_Implementation.java
$ java Interface_Implementation
```

```
Enter length:6
Enter breadth:7
Area:42
```

3.Java Program to Convert Integer Values into Binary

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Integer Values into Binary.

Enter any decimal number as an input. After that we operatons like modulo and division to convert the gven input into binary number.

Here is the source code of the Java Program to Convert Integer Values into Binary. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Decimal_Binary
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, m;
7.         String x = "";
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter the Decimal Number:");
10.        n = s.nextInt();
11.        while(n > 0)
12.        {
13.            int a = n % 2;
14.            x = a + x;
15.            n = n / 2;
16.        }
17.        System.out.println(x);
18.    }
19. }
```

Output:

```
$ javac Decimal_Binary.java
$ java Decimal_Binary
```

```
Enter the Decimal Number:19
10011
```

4.Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Without Using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Without Using Recursion. Gray code is a binary numeral system where two successive values differ in only one bit.

Enter any binary number as an input. After that we perform operations like modulo and division to convert it into gray code.

Here is the source code of the Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Without Using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import static java.lang.StrictMath.pow;
2. import java.util.Scanner;
3. public class Binary_Gray
4. {
5.     public static void main(String[] args)
6.     {
7.         int a, b, x, result = 0, i = 0;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter Binary number:");
10.        x = s.nextInt();
11.        while(x != 0)
12.        {
13.            a = x % 10;
14.            x = x / 10;
15.            b = x % 10;
16.            if((a & ~b) == 1 || (~a & b) == 1)
17.            {
18.                result = (int) (result + pow(10,i));
19.            }
20.            i++;
21.        }
22.        System.out.println("Gray Code:" + result);
23.    }
24.}
```

Output:

```
$ javac Binary_Gray.java
$ java Binary_Gray
```

```
Enter Binary number:1001
Gray Code:1101
```

5.Java Program to Find Sum of N Numbers using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Find Sum of N Numbers using Recursion.

Enter the number of elements you want and then enter all the required elements as an input. Now we pass all the elements along with the length of array and a zero to new function where we find sum of all the given numbers with the help of recursion.

Here is the source code of the Java Program to Find Sum of N Numbers using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Sum_Numbers
3. {
4.     int sum = 0, j = 0;
5.     public static void main(String[] args)
6.     {
7.         int n;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter the no. of elements you want:");
10.        n = s.nextInt();
11.        int a[] = new int[n];
12.        System.out.print("Enter all the elements you want:");
13.        for(int i = 0; i < n; i++)
14.        {
15.            a[i] = s.nextInt();
16.        }
17.        Sum_Numbers obj = new Sum_Numbers();
18.        int x = obj.add(a, a.length, 0);
19.        System.out.println("Sum:" + x);
20.    }
21.    int add(int a[], int n, int i)
22.    {
23.        if(i < n)
24.        {
25.            return a[i] + add(a, n, ++i);
26.        }
27.        else
28.        {
29.            return 0;
30.        }
31.    }
32.}
33.Output:
34.<pre lang="text" cssfile="hk1_style">
35.$ javac Sum_Numbers.java
36.$ java Sum_Numbers
37.
38.Enter the no. of elements you want:6
39.Enter all the elements you want:2
40.3
41.5
42.6
43.7
44.1
45.Sum:24
```

6.Java Program to Find Sum of Digits of a Number using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Find Sum of Digits of a Number using Recursion.

Enter any Decimal number as an input. Now we pass that number to a new function where we use modulo operator to find sum of digits as output along with the help of recursion.

Here is the source code of the Java Program to Find Sum of Digits of a Number using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Digit_Sum
3. {
4.     int sum = 0;
5.     public static void main(String[] args)
6.     {
7.         int n;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter the number:");
10.        n = s.nextInt();
11.        Digit_Sum obj = new Digit_Sum();
12.        int a = obj.add(n);
13.        System.out.println("Sum:"+a);
14.    }
15.    int add(int n)
16.    {
17.        sum = n % 10;
18.        if(n == 0)
19.        {
20.            return 0;
21.        }
22.        else
23.        {
24.            return sum + add(n / 10);
25.        }
26.
27.    }
28.}
```

Output:

```
$ javac Digit_Sum.java
$ java Digit_Sum
```

```
Enter the number:345
Sum:12
```

7.Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Using Recursion. Gray code is a binary numeral system where two successive values differ in only one bit.

Enter any binary number as an input. Now we pass the given number along with zero to different function where with the help of different operations like modulo, division and recursion we get the gray code as an output.

Here is the source code of the Java Program to Convert Binary Code of a Number into its Equivalent Gray's Code Using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import static java.lang.StrictMath.pow;
2. import java.util.Scanner;
3. public class Binary_Gray_Recursion
4. {
5.     public static void main(String[] args)
6.     {
7.         int n, result = 0;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter Binary number:");
10.        n = s.nextInt();
11.        Binary_Gray_Recursion obj = new Binary_Gray_Recursion();
12.        result = obj.GrayCode(n, 0);
13.        System.out.println("Gray Code:" + result);
14.    }
15.    int GrayCode(int x,int i)
16.    {
17.        int a, b, result = 0;
18.        if(x != 0)
19.        {
20.            a = x % 10;
21.            x = x / 10;
22.            b = x % 10;
23.            if((a & ~b) == 1 || (~a & b) == 1)
24.            {
25.                result = (int) (result + pow(10,i));
26.            }
27.            return GrayCode(x, ++i) + result;
28.        }
29.        return 0;
30.    }
31.}
```

Output:

```
$ javac Binary_Gray_Recursion.java
$ java Binary_Gray_Recursion
```

```
Enter Binary number:1001
Gray Code:1101
```

8.Java Program to Illustrate the Use of Various Bitwise Operators

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate the Use of Various Bitwise Operators.

Enter any two decimal numbers as an input. After that select any option from different available options to perform various bitwise operations and get the desired output.

Here is the source code of the Java Program to Illustrate the Use of Various Bitwise Operators. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Bitwise_Operation
3. {
4.     public static void main(String[] args)
5.     {
6.         int m, n, x, a;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter First number:");
9.         m = s.nextInt();
10.        System.out.print("Enter Second number:");
11.        n = s.nextInt();
12.        while(true)
13.        {
14.            System.out.println("");
15.            System.out.println("Press 1 for Right Shift by 2:");
16.            System.out.println("Press 2 for Left Shift by 2:");
17.            System.out.println("Press 3 for Bitwise AND:");
18.            System.out.println("Press 4 for Bitwise OR by 2:");
19.            System.out.println("Press 5 for Bitwise Exclusive OR:");
20.            System.out.println("Press 6 for Bitwise NOT:");
21.            System.out.println("Press 7 to Exit:");
22.            System.out.println("");
23.            System.out.print("Option:");
24.            x = s.nextInt();
25.            switch(x)
26.            {
27.                case 1:
28.                    a = m << 2;
29.                    System.out.println("Result after left shift by 2:"+a);
30.                    break;
31.
32.                case 2:
33.                    a = n >> 2;
34.                    System.out.println("Result after right shift by 2:"+a);
35.                    break;
36.
37.                case 3:
38.                    a = m & n;
39.                    System.out.println("Result after bitwise AND:"+a);
40.                    break;
41.
42.                case 4:
43.                    a = m | n;
44.                    System.out.println("Result after bitwise OR:"+a);
45.                    break;
46.
47.                case 5:
48.                    a = m ^ n;
49.                    System.out.println("Result after bitwise Exclusive OR:"+a);
```

```
50.     break;
51.
52.     case 6:
53.         a = ~ m;
54.         System.out.println("Result after bitwise NOT:"+a);
55.         break;
56.
57.     case 7:
58.         System.exit(0);
59.     }
60. }
61. }
62. }
```

Output:

```
$ javac Bitwise_Operation.java
$ java Bitwise_Operation
```

Enter First number:5

Enter Second number:6

Press 1 for Right Shift by 2:

Press 2 for Left Shift by 2:

Press 3 for Bitwise AND:

Press 4 for Bitwise OR by 2:

Press 5 for Bitwise Exclusive OR:

Press 6 for Bitwise NOT:

Press 7 to Exit:

Option:3

Result after bitwise AND:4

Press 1 for Right Shift by 2:

Press 2 for Left Shift by 2:

Press 3 for Bitwise AND:

Press 4 for Bitwise OR by 2:

Press 5 for Bitwise Exclusive OR:

Press 6 for Bitwise NOT:

Press 7 to Exit:

Option:7

9.Java Program to Count the Number of Bits set to One

[« Prev](#)

[Next »](#)

This is a Java Program to Count the Number of Bits set to One.

Enter any decimal number as an input. After that we first convert the given decimal number into binary number and then check at every position for 1. If 1 occurs then we increase the counter by one. Hence at last we get the counter value as the output.

Here is the source code of the Java Program to Count the Number of Bits set to One. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Count_One
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, m, count = 0;
7.         String x = "";
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter the Decimal Number:");
10.        n = s.nextInt();
11.        while(n > 0)
12.        {
13.            int a = n % 2;
14.            x = a + x;
15.            n = n / 2;
16.        }
17.        int l = x.length();
18.        for(int i = 0; i < l; i++)
19.        {
20.            if(x.charAt(i) == '1')
21.            {
22.                count++;
23.            }
24.        }
25.        System.out.println("No. of 1's are:" + count);
26.    }
27.}
```

Output:

```
$ javac Count_One.java
$ java Count_One
```

```
Enter the Decimal Number:15
```

```
No. of 1's are:4
```

10. Java Program to Check if a given Bit Position is set to One or not

[« Prev](#)

[Next »](#)

This is a Java Program to Check if a given Bit Position is set to One or not.

Enter any decimal number as an input. Also mention the bit position at which you want to check whether one is present or not. After that we first convert the given decimal number into binary number and then check the element at that given position and generate output.

Here is the source code of the Java Program to Check if a given Bit Position is set to One or not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Bit_Postion
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, m;
7.         String x = "";
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter any Decimal Number:");
10.        n = s.nextInt();
11.        while(n > 0)
12.        {
13.            int a = n % 2;
14.            x = a + x;
15.            n = n / 2;
16.        }
17.        System.out.print("Enter the position where you want to check:");
18.        int l = x.length();
19.        m = s.nextInt();
20.        if((l - m) >= 0 && (x.charAt(l - m) == '1'))
21.        {
22.            System.out.println("1 is present at given bit position");
23.        }
24.        else
25.        {
26.            System.out.println("0 is present at given bit position");
27.        }
28.    }
29.}
```

Output:

```
$ javac Bit_Postion.java
$ java Bit_Postion
```

```
Enter any Decimal Number:6
Enter the position where you want to check:2
1 is present at given bit position
```

11. Java Program to Print Binary Equivalent of an Integer using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Print Binary Equivalent of an Integer using Recursion.

Enter any integer number as an input. After that we pass the given number to the other function where by using modulo operator along with recursion we get the reverse of a number as an output.

Here is the source code of the Java Program to Print Binary Equivalent of an Integer using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Binary_Recursion
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number:");
9.         n = s.nextInt();
10.        Binary_Recursion obj = new Binary_Recursion();
11.        String m = obj.Binary(n);
12.        System.out.println("Answer:"+m);
13.    }
14.    String Binary(int x)
15.    {
16.        if(x > 0)
17.        {
18.            int a = x % 2;
19.            x = x / 2;
20.            return a + "" + Binary(x);
21.        }
22.        return "";
23.    }
24.}
```

Output:

```
$ javac Binary_Recursion.java
$ java Binary_Recursion
```

```
Enter the number:19
Answer:11001
```

12. Java Program to Find the Biggest of 3 Numbers

[« Prev](#)

[Next »](#)

This is a Java Program to Find the Biggest of 3 Numbers.

Enter any three integer numbers as an input. Now we check the first number against the second and third number. If it false then we check for second number against third. If it is also false then accordingly third number will be declared the largest number of the given three numbers.

Here is the source code of the Java Program to Find the Biggest of 3 Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Biggest_Number
3. {
4.     public static void main(String[] args)
5.     {
6.         int x, y, z;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the first number:");
9.         x = s.nextInt();
10.        System.out.print("Enter the second number:");
11.        y = s.nextInt();
12.        System.out.print("Enter the third number:");
13.        z = s.nextInt();
14.        if(x > y && x > z)
15.        {
16.            System.out.println("Largest number is:"+x);
17.        }
18.        else if(y > z)
19.        {
20.            System.out.println("Largest number is:"+y);
21.        }
22.        else
23.        {
24.            System.out.println("Largest number is:"+z);
25.        }
26.
27.    }
28.}
```

Output:

```
$ javac Biggest_Number.java
$ java Biggest_Number
```

```
Enter the first number:10
Enter the second number:17
Enter the third number:15
Largest number is:17
```

13. Java Program to Calculate the Sum of Odd & Even Numbers

[« Prev](#)

[Next »](#)

This is a Java Program to Calculate the Sum of Odd & Even Numbers.

Enter the number of elements you want in array. Now enter all the elements you want in that array. We begin from the first element and check if it is odd or even. Hence we add that number into the required addition list depending whether it is even or odd.

Here is the source code of the Java Program to Calculate the Sum of Odd & Even Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Sum_Odd_Even
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, sumE = 0, sumO = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number of elements in array:");
9.         n = s.nextInt();
10.        int[] a = new int[n];
11.        System.out.println("Enter the elements of the array:");
12.        for(int i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        for(int i = 0; i < n; i++)
17.        {
18.            if(a[i] % 2 == 0)
19.            {
20.                sumE = sumE + a[i];
21.            }
22.            else
23.            {
24.                sumO = sumO + a[i];
25.            }
26.        }
27.        System.out.println("Sum of Even Numbers:"+sumE);
28.        System.out.println("Sum of Odd Numbers:"+sumO);
29.    }
30.}
```

Output:

```
$ javac Sum_Odd_Even.java
$ java Sum_Odd_Even
```

Enter the number of elements in array:6

Enter the elements of the array:

```
1
3
2
6
7
9
```

Sum of Even Numbers:8

Sum of Odd Numbers:20

14. Java Program to Check if a Given Integer is Odd or Even

[« Prev](#)

[Next »](#)

This is a Java Program to Check if a Given Integer is Odd or Even.

Enter any integer number as an input. Now we check its remainder with modulo operator by two. If remainder is zero the given number is even. If remainder is 1 then the given number is odd. Hence we show output according to the remainder

Here is the source code of the Java Program to Check if a Given Integer is Odd or Even. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Odd_Even
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number you want to check:");
9.         n = s.nextInt();
10.        if(n % 2 == 0)
11.        {
12.            System.out.println("The given number "+n+" is Even ");
13.        }
14.        else
15.        {
16.            System.out.println("The given number "+n+" is Odd ");
17.        }
18.    }
19. }
```

Output:

```
$ javac Odd_Even.java
$ java Odd_Even
```

```
Enter the number you want to check:15
The given number 15 is Odd
```

15. Java Program to Check if a Given Integer is Positive or Negative

[« Prev](#)

[Next »](#)

This is a Java Program to Check if a Given Integer is Positive or Negative.

Enter any integer number as an input. Now we check whether the given number is greater than zero or not. If the given number is greater than zero than it is positive. If it is less than zero than it is negative and if it is zero than it is neither positive nor negative.

Here is the source code of the Java Program to Check if a Given Integer is Positive or Negative. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Postive_Negative
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number you want to check:");
9.         n = s.nextInt();
10.        if(n > 0)
11.        {
12.            System.out.println("The given number "+n+" is Positive");
13.        }
14.        else if(n < 0)
15.        {
16.            System.out.println("The given number "+n+" is Negative");
17.        }
18.        else
19.        {
20.            System.out.println("The given number "+n+" is neither Positive nor Negative ");
21.        }
22.    }
23.}
```

Output:

```
$ javac Postive_Negative.java
$ java Postive_Negative
```

```
Enter the number you want to check:6
The given number 6 is Positive
```

16. Java Program to Reverse a Given Number

« [Prev](#)

[Next »](#)

This is a Java Program to Reverse a Given Number.

Enter any integer number as an input. After that we use modulus and division operation respectively to generate output.

Here is the source code of the Java Program to Reverse a Given Number. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Reverse_Number
3. {
4.     public static void main(String args[])
5.     {
6.         int m, n, sum = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the number:");
9.         m = s.nextInt();
10.        while(m > 0)
11.        {
12.            n = m % 10;
13.            sum = sum * 10 + n;
14.            m = m / 10;
15.        }
16.        System.out.println("Reverse:"+sum);
17.    }
18. }
```

Output:

```
$ javac Reverse_Number.java
$ java Reverse_Number
```

```
Enter the number:567
Reverse:765
```

17. Java Program to Read Two Integers M and N & Swap their Values

[« Prev](#)

[Next »](#)

This is a Java Program to Read Two Integers M and N & Swap their Values.

Enter any two integer numbers as input. After that we take a new variable temp and copy first variable to this temp variable. Now we copy second variable to first variable and temp variable into second variable. Hence we get the swapped values as output.

Here is the source code of the Java Program to Read Two Integers M and N & Swap their Values. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Swap_Integers
3. {
4.     public static void main(String args[])
5.     {
6.         int m, n, temp;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the first number:");
9.         m = s.nextInt();
10.        System.out.print("Enter the second number:");
11.        n = s.nextInt();
12.        temp = m;
13.        m = n;
14.        n = temp;
15.        System.out.println("After Swapping");
16.        System.out.println("First number:"+m);
17.        System.out.println("Second number:"+n);
18.    }
19.}
```

Output:

```
$ javac Swap_Integers.java
$ java Swap_Integers
```

```
Enter the first number:5
Enter the second number:7
After Swapping
First number:7
Second number:5
```

18. Java Program to Illustrate how User Authentication is Done

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate how User Authentication is Done.

Enter username and password as input strings. After that we match both strings against given username and password. If it matches then Authentication is successfull or else Authentication fails.

Here is the source code of the Java Program to Illustrate how User Authentication is Done. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class User_Authentication
3. {
4.     public static void main(String args[])
5.     {
6.         String username, password;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter username:");//username:user
9.         username = s.nextLine();
10.        System.out.print("Enter password:");//password:user
11.        password = s.nextLine();
12.        if(username.equals("user") && password.equals("user"))
13.        {
14.            System.out.println("Authentication Successful");
15.        }
16.        else
17.        {
18.            System.out.println("Authentication Failed");
19.        }
20.    }
21.}
```

Output:

```
$ javac User_Authentication.java
$ java User_Authentication
```

```
Enter username:user
Enter password:user
Authentication Successful
```

```
Enter username:abcd
Enter password:1234
Authentication Failed
```

19. Java Program to Swap the Contents of two Numbers using Bitwise XOR Operation

[« Prev](#)

[Next »](#)

This is a Java Program to Swap the Contents of two Numbers using Bitwise XOR Operation.

Enter any two integer number as input. After that we first find bitwise XOR of first and second variables and store it in first variable. Now again we perform bitwise XOR operation of the same variables and store it in second variable. Finally bitwise XOR is performed on first and second variable and result is stored in first variable. Hence we get the swapped value as an output.

Here is the source code of the Java Program to Swap the Contents of two Numbers using Bitwise XOR Operation. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Swap_BitwiseXOR
3. {
4.     public static void main(String args[])
5.     {
6.         int m, n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter the first number:");
9.         m = s.nextInt();
10.        System.out.print("Enter the second number:");
11.        n = s.nextInt();
12.        m = m ^ n;
13.        n = m ^ n;
14.        m = m ^ n;
15.        System.out.println("After Swapping");
16.        System.out.println("First number:"+m);
17.        System.out.println("Second number:"+n);
18.    }
19.}
```

Output:

```
$ javac Swap_BitwiseXOR.java
$ java Swap_BitwiseXOR
```

```
Enter the first number:6
Enter the second number:7
After Swapping
First number:7
Second number:6
```

20. Java Program to Find Largest Between Three Numbers Using Ternary Operator

[« Prev](#)

[Next »](#)

This is a Java Program to Find Largest Between Three Numbers Using Ternary Operator.

Enter any three integer numbers as an input from which you want to find the largest integer. After that we use ternary operator(?, >, :) to find largest number as the output.

Here is the source code of the Java Program to Find Largest Between Three Numbers Using Ternary Operator. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Largest_Ternary
3. {
4.     public static void main(String[] args)
5.     {
6.         int a, b, c, d;
7.         Scanner s = new Scanner(System.in);
8.         System.out.println("Enter all three numbers:");
9.         a = s.nextInt();
10.        b = s.nextInt();
11.        c = s.nextInt();
12.        d = c > (a > b ? a : b) ? c : ((a > b) ? a : b);
13.        System.out.println("Largest Number:"+d);
14.    }
15.}
```

Output:

```
$ javac Largest_Ternary.java
$ java Largest_Ternary
```

Enter all three numbers:

```
5
6
7
```

Largest Number:7

21. Java Program to Check if given Alphabets are Uppercase or Lowercase or Digits

[« Prev](#)

[Next »](#)

This is a Java Program to Check if given Alphabets are Uppercase or Lowercase or Digits.

Enter any character as input. After that we match the ASCII value of that character against the given three cases. If that value matches then we get the output accordingly.

Here is the source code of the Java Program to Check if given Alphabets are Uppercase or Lowercase or Digits. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.io.BufferedReader;
2. import java.io.IOException;
3. import java.io.InputStreamReader;
4. public class Alphabet_Check
5. {
6.     public static void main(String args[]) throws IOException
7.     {
8.         char m;
9.         BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
10.        System.out.print("Enter any alphabet:");
11.        m = (char) bf.read();
12.        if(m >= 97 && m <= 123)
13.        {
14.            System.out.println("Lower Case");
15.        }
16.        else if(m >= 65 && m <= 96)
17.        {
18.            System.out.println("Upper Case");
19.        }
20.        else if(m >= 48 && m <= 57)
21.        {
22.            System.out.println("Digit");
23.        }
24.    }
25. }
```

Output:

```
$ javac Alphabet_Check.java
$ java Alphabet_Check
```

```
Enter any alphabet:B
Upper Case
```

```
Enter any alphabet:z
Lower Case
```

```
Enter any alphabet:9
Digit
```

22. Java Program to Find Out the Number of Objects Created of a Class

[« Prev](#)

[Next »](#)

This is a Java Program to Find Out the Number of Objects Created of a Class.

Whenever an object is made of a class, its constructor is invoked. Whenever the constructor runs we increment the counter value. Hence number of objects created of a class equals to the counter value.

Here is the source code of the Java Program to Find Out the Number of Objects Created of a Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Number_Objects
2. {
3.     static int count=0;
4.     Number_Objects()
5.     {
6.         count++;
7.     }
8.     public static void main(String[] args)
9.     {
10.        Number_Objects obj1 = new Number_Objects();
11.        Number_Objects obj2 = new Number_Objects();
12.        Number_Objects obj3 = new Number_Objects();
13.        Number_Objects obj4 = new Number_Objects();
14.        System.out.println("Number of objects created:"+count);
15.    }
16.}
```

Output:

```
$ javac Number_Objects.java
$ java Number_Objects
```

```
Number of objects created:4
```

23. Java Program to Print Multiplication Table for any Number

[« Prev](#)

[Next »](#)

This is a Java Program to Print Multiplication Table for any Number.

Enter any integer number as input of which you want multiplication table. After that we use for loop from one to ten to generate multiplication of that number.

Here is the source code of the Java Program to Print Multiplication Table for any Number. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Multiplication_Table
3. {
4.     public static void main(String[] args)
5.     {
6.         Scanner s = new Scanner(System.in);
7.         System.out.print("Enter number:");
8.         int n=s.nextInt();
9.         for(int i=1; i <= 10; i++)
10.        {
11.            System.out.println(n+" * "+i+" = "+n*i);
12.        }
13.    }
14.}
```

Output:

```
$ javac Multiplication_Table.java
$ java Multiplication_Table
```

Enter number:7

```
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

24. Java Program to Accept Array Elements and Calculate Sum

[« Prev](#)

[Next »](#)

This is a Java Program to Accept Array Elements and Calculate Sum. An array is a group of like-typed variables that are referred to by a common name. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index.

Enter the size of array and then enter all the elements of that array. Now we calculate the sum of elements of the array with the help of for loop.

Here is the source code of the Java Program to Accept Array Elements and Calculate Sum. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Array_Sum
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, sum = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for(int i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.            sum = sum + a[i];
16.        }
17.        System.out.println("Sum:" +sum);
18.    }
19.}
```

Output:

```
$ javac Array_Sum.java
$ java Array_Sum
```

Enter no. of elements you want in array:5

Enter all the elements:

```
1
2
3
4
5
Sum:15
```

25. Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage

[« Prev](#)

[Next »](#)

This is a Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage. A one-dimensional array is, essentially, a list of like-typed variables.

Enter the number of subjects and then enter marks if students in all those subjects. Now we us for loop to calculate total marks of the student and hence divide it by the total number of subjects to get percentage marks.

Here is the source code of the Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Student_Marks
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, total = 0, percentage;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of subject:");
9.         n = s.nextInt();
10.        int marks[] = new int[n];
11.        System.out.println("Enter marks out of 100:");
12.        for(int i = 0; i < n; i++)
13.        {
14.            marks[i] = s.nextInt();
15.            total = total + marks[i];
16.        }
17.        percentage = total / n;
18.        System.out.println("Sum:"+total);
19.        System.out.println("Percentage:"+percentage);
20.    }
21.}
```

Output:

```
$ javac Student_Marks.java
$ java Student_Marks
```

```
Enter no. of subject:5
Enter marks out of 100:
86
89
91
82
78
Sum:426
Percentage:85
```

26. Java Program to Search Key Elements in an Array

[« Prev](#)

[Next »](#)

This is a Java Program to Search Key Elements in an Array.

Enter the size of array and then enter all the elements of that array. Now enter the element you want to search for. With the help of for loop we can find out the location of the element easily.

Here is the source code of the Java Program to Search Key Elements in an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Search_Element
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, x, flag = 0, i = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for(i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        System.out.print("Enter the element you want to find:");
17.        x = s.nextInt();
18.        for(i = 0; i < n; i++)
19.        {
20.            if(a[i] == x)
21.            {
22.                flag = 1;
23.                break;
24.            }
25.            else
26.            {
27.                flag = 0;
28.            }
29.        }
30.        if(flag == 1)
31.        {
32.            System.out.println("Element found at position:"+(i + 1));
33.        }
34.        else
35.        {
36.            System.out.println("Element not found");
37.        }
38.    }
39.}
```

Output:

```
$ javac Search_Element.java
$ java Search_Element
```

```
Enter no. of elements you want in array:7
```

```
Enter all the elements:
```

```
2
4
```

1

5

7

6

9

Enter the element you want to find:5

Element found at position:4

27. Java Program to Count the Number of Occurrence of an Element in an Array

[« Prev](#)

[Next »](#)

This is a Java Program to Count the Number of Occurrence of an Element in an Array.

Enter size of array and then enter all the elements of that array. Now enter the element of which you want to count occurrences. With the help of for loop now we can easily calculate number of occurrences of the given element.

Here is the source code of the Java Program to Count the Number of Occurrence of an Element in an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Count_Occurrence
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, x, count = 0, i = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for(i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        System.out.print("Enter the element of which you want to count number of occurrences:");
17.        x = s.nextInt();
18.        for(i = 0; i < n; i++)
19.        {
20.            if(a[i] == x)
21.            {
22.                count++;
23.            }
24.        }
25.        System.out.println("Number of Occurrence of the Element:" + count);
26.    }
27.}
```

Output:

```
$ javac Count_Occurrence.java
$ java Count_Occurrence
```

Enter no. of elements you want in array:5

Enter all the elements:

```
2
3
3
4
3
```

Enter the element of which you want to count number of occurrences:3
Number of Occurrence of the Element:3

28. Java Program to Find if a given Integer X appears more than N/2 times in a Sorted Array of N Integers

[« Prev](#)

[Next »](#)

This is a Java Program to Find if a given Integer X appears more than N/2 times in a Sorted Array of N Integers. Enter size of array and then enter all the elements of that array. Now enter the element you want to check. With the help of for loop now we check if it occurs more than n/2 times or not.

Here is the source code of the Java Program to Find if a given Integer X appears more than N/2 times in a Sorted Array of N Integers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Appear_Array
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, x, count = 0, i = 0, temp = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for(i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        for(i = 0; i < n; i++)
17.        {
18.            for(int j = i + 1; j < n; j++)
19.            {
20.                if(a[i] > a[j])
21.                {
22.                    temp = a[i];
23.                    a[i] = a[j];
24.                    a[j] = temp;
25.                }
26.            }
27.        }
28.        System.out.print("Enter the element which you want to check:");
29.        x = s.nextInt();
30.        for(i = 0; i < n; i++)
31.        {
32.            if(a[i] == x)
33.            {
34.                count++;
35.            }
36.        }
37.        if(count > (n / 2))
38.        {
39.            System.out.println("Given Integer appears more than N/2 times");
40.        }
41.        else
42.        {
43.            System.out.println("Given Integer does not appear more than N/2 times");
44.        }
45.    }
46.}
```

Output:

```
$ javac Appear_Array.java  
$ java Appear_Array
```

Enter no. of elements you want in array:5

Enter all the elements:

```
2  
4  
1  
3  
2
```

Enter the element which you want to check:2

Given Integer does not appear more than N/2 times

29. Java Program to Increment Every Element of the Array by One & Print Incremented Array

[« Prev](#)

[Next »](#)

This is a Java Program to Increment Every Element of the Array by One & Print Incremented Array.

Enter size of array and then enter all the elements of that array. Now using for loop we increment all the elements of the array.

Here is the source code of the Java Program to Increment Every Element of the Array by One & Print Incremented Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Increment_Array
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, i = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for(i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.            a[i]++;
16.        }
17.        System.out.print("Elements of array after increment by 1:");
18.        for(i = 0; i < n - 1; i++)
19.        {
20.            System.out.print(a[i]+",");
21.        }
22.        System.out.print(a[n-1]);
23.    }
24.}
```

Output:

```
$ javac Increment_Array.java
$ java Increment_Array
```

Enter no. of elements you want in array:5

Enter all the elements:

```
2
5
8
6
9
```

Elements of array after increment by 1:3,6,9,7,10

30. Java Program to Use Strings in Switch Statements

[« Prev](#)

[Next »](#)

This is a Java Program to Use Strings in Switch Statements.

Enter the string as input. Now we match the given string with different cases and then generate the output accordingly.

Here is the source code of the Java Program to Use Strings in Switch Statements. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Strings_Switch
3. {
4.     public static void main(String[] args)
5.     {
6.         String week;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter choice:");
9.         week = s.nextLine();
10.        switch (week)
11.        {
12.            case "Monday":
13.                System.out.print("Day: Monday");
14.                break;
15.
16.            case "Tuesday":
17.                System.out.print("Day: Tuesday");
18.                break;
19.
20.            case "Wednesday":
21.                System.out.print("Day: Wednesday");
22.                break;
23.
24.            case "Thursday":
25.                System.out.print("Day: Thursday");
26.                break;
27.
28.            case "Friday":
29.                System.out.print("Day: Friday");
30.                break;
31.
32.            case "Saturday":
33.                System.out.print("Day: Saturday");
34.                break;
35.
36.            case "Sunday":
37.                System.out.print("Day: Sunday");
38.                break;
39.        }
40.    }
41.}
```

Output:

```
$ javac Strings_Switch.java
$ java Strings_Switch
```

```
Enter choice:Saturday
Day: Saturday
```

31. Java Program to Return a Value from a Method

[« Prev](#)

[Next »](#)

This is a Java Program to Return a Value from a Method.

We make a method named input which is used to get input from the user. We also make a method named add which is used to perform addition and return the result back to input method where we finally print the result.

Here is the source code of the Java Program to Return a Value from a Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Return_Value
3. {
4.     void input()
5.     {
6.         int x, y, z;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter first integer:");
9.         x = s.nextInt();
10.        System.out.print("Enter second integer:");
11.        y = s.nextInt();
12.        z = add(x, y);
13.        System.out.println("Result:"+z);
14.    }
15.    int add(int a, int b)
16.    {
17.        int c;
18.        c = a + b;
19.        return c;
20.    }
21.    public static void main(String[] args)
22.    {
23.        Return_Value obj = new Return_Value();
24.        obj.input();
25.    }
26.}
```

Output:

```
$ javac Return_Value.java
$ java Return_Value
```

```
Enter first integer:5
Enter second integer:6
Result:11
```

32. Java Program to Check Whether Given Number is Divisible by 5

[« Prev](#)

[Next »](#)

This is a Java Program to Check Whether Given Number is Divisible by 5.

Enter any integer as an input. We use modulus operation to check the divisiblity of given integer by 5. If the given integer gives zero as remainder when divided by 5 then it is divisible by 5 or else its not divisible by 5.

Here is the source code of the Java Program to Check Whether Given Number is Divisible by 5. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Check_Divisiblity
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any number:");
9.         n = s.nextInt();
10.        if(n % 5 == 0)
11.        {
12.            System.out.println(n+" is divisible by 5");
13.        }
14.        else
15.        {
16.            System.out.println(n+" is not divisible by 5");
17.        }
18.    }
19. }
```

Output:

```
$ javac Check_Divisiblity.java
$ java Check_Divisiblity
```

```
Enter any number:45
45 is divisible by 5
```

```
Enter any number:16
16 is not divisible by 5
```

33. Java Program to Find if a Given Year is a Leap Year

[« Prev](#)

[Next »](#)

This is a Java Program to Find if a Given Year is a Leap Year.

Enter any year as an input. We first check whether the given year is divisible by 400 or not. If it is divisible then it is a leap year else we check for further conditions. Now if it is divisible by 100 then it is not a leap year or else we further divide it by 4. If it is divisible then it is a leap year else its not.

Here is the source code of the Java Program to Find if a Given Year is a Leap Year. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Check_Leap_Year
3. {
4.     public static void main(String args[])
5.     {
6.         Scanner s = new Scanner(System.in);
7.         System.out.print("Enter any year:");
8.         int year = s.nextInt();
9.         boolean flag = false;
10.        if(year % 400 == 0)
11.        {
12.            flag = true;
13.        }
14.        else if (year % 100 == 0)
15.        {
16.            flag = false;
17.        }
18.        else if(year % 4 == 0)
19.        {
20.            flag = true;
21.        }
22.        else
23.        {
24.            flag = false;
25.        }
26.        if(flag)
27.        {
28.            System.out.println("Year "+year+" is a Leap Year");
29.        }
30.        else
31.        {
32.            System.out.println("Year "+year+" is not a Leap Year");
33.        }
34.    }
35.}
```

Output:

```
$ javac Check_Leap_Year.java
$ java Check_Leap_Year
```

```
Enter any year:1800
Year 1800 is not a Leap Year
```

```
Enter any year:2000
Year 2000 is a Leap Year
```

34. Java Program to Display Numbers from 1 to 10 Using For Loop

[« Prev](#)

[Next »](#)

This is a Java Program to Display Numbers from 1 to 10 Using For Loop.

We use For Loop in which we initialise a variable to 1 and increments each time by 1 till we reach 10. The loop breaks when variable attains value 11.

Here is the source code of the Java Program to Display Numbers from 1 to 10 Using For Loop. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Use_For_Loop
2. {
3.     public static void main(String[] args)
4.     {
5.         for(int i = 1; i <= 10; i++)
6.         {
7.             System.out.println(i);
8.         }
9.     }
10.}
```

Output:

```
$ javac Use_For_Loop.java
$ java Use_For_Loop
```

```
1
2
3
4
5
6
7
8
9
10
```

35. Java Program to Display the ATM Transaction

[« Prev](#)

[Next »](#)

This is a Java Program to Display the ATM Transaction.

The user will choose from any one of the available options as input. Different cases using switch case have been provided for different operations like withdraw, deposit and check balance.

Here is the source code of the Java Program to Display the ATM Transaction. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class ATM_Transaction
3. {
4.     public static void main(String args[])
5.     {
6.         int balance = 5000, withdraw, deposit;
7.         Scanner s = new Scanner(System.in);
8.         while(true)
9.         {
10.             System.out.println("Automated Teller Machine");
11.             System.out.println("Choose 1 for Withdraw");
12.             System.out.println("Choose 2 for Deposit");
13.             System.out.println("Choose 3 for Check Balance");
14.             System.out.println("Choose 4 for EXIT");
15.             System.out.print("Choose the operation you want to perform:");
16.             int n = s.nextInt();
17.             switch(n)
18.             {
19.                 case 1:
20.                     System.out.print("Enter money to be withdrawn:");
21.                     withdraw = s.nextInt();
22.                     if(balance >= withdraw)
23.                     {
24.                         balance = balance - withdraw;
25.                         System.out.println("Please collect your money");
26.                     }
27.                     else
28.                     {
29.                         System.out.println("Insufficient Balance");
30.                     }
31.                     System.out.println("");
32.                     break;
33.
34.                 case 2:
35.                     System.out.print("Enter money to be deposited:");
36.                     deposit = s.nextInt();
37.                     balance = balance + deposit;
38.                     System.out.println("Your Money has been successfully deposited");
39.                     System.out.println("");
40.                     break;
41.
42.                 case 3:
43.                     System.out.println("Balance : "+balance);
44.                     System.out.println("");
45.                     break;
46.
47.                 case 4:
48.                     System.exit(0);
49.             }
50.         }
51.     }
52. }
```

Output:

```
$ javac ATM_Transaction.java  
$ java ATM_Transaction
```

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:1
Enter money to be withdrawn:2000
Please collect your money

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:3
Balance : 3000

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:4

36. Java Program to Print any Statement without using Semicolon

[« Prev](#)

[Next »](#)

This is a Java Program to Print any Statement without using Semicolon.

We use statements like if and for to print any statement without using semicolon. Since these statements do not require semicolons at the end of these statements, we use them to get desired output.

Here is the source code of the Java Program to Print any Statement without using Semicolon. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Print_Without_Semicolon
2. {
3.     public static void main(String[] args)
4.     {
5.         int i = 0;
6.         if(System.out.printf("Sanfoundry!! ") == null) {}
7.         for(i = 1; i < 2; System.out.println("Hello World."))
8.         {
9.             i++;
10.        }
11.    }
12.}
```

Output:

```
$ javac Print_Without_Semicolon.java
$ java Print_Without_Semicolon
```

```
Sanfoundry!! Hello World.
```

37. Java Program to Display the IP Address of the System

[« Prev](#)

[Next »](#)

This is a Java Program to Display the IP Address of the System.

InetAddress class represents an Internet Protocol (IP) address. We use the method named getHostAddress() of this class to get the IP address string in textual presentation.

Here is the source code of the Java Program to Display the IP Address of the System. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.net.InetAddress;
2. public class IP_Address
3. {
4.     public static void main(String args[]) throws Exception
5.     {
6.         InetAddress IP = InetAddress.getLocalHost();
7.         System.out.println("IP of my system is := "+IP.getHostAddress());
8.     }
9. }
```

Output:

```
$ javac IP_Address.java
$ java IP_Address
```

```
IP of my system is := 127.0.1.1
```

38. Java Program to Print a Semicolon without using a Semicolon anywhere in the Code

[« Prev](#)

[Next »](#)

This is a Java Program to Print a Semicolon without using a Semicolon anywhere in the Code.

We take a integer variable and assign it a value of fifty nine. Now with the help Implicit Casting we convert this integer value into char and since ASCII value of semicolon(;) is fifty nine we get the output as semicolon.

Here is the source code of the Java Program to Print a Semicolon without using a Semicolon anywhere in the Code. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Semicolon
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         char a;
8.         n = 59;
9.         a = (char) n;
10.        System.out.println(a);
11.    }
12.}
```

Output:

```
$ javac Semicolon.java
$ java Semicolon
```

```
;
```

39. Java Program to Increment by 1 All the Digits of a given Integer

[« Prev](#)

[Next »](#)

This is a Java Program to Increment by 1 All the Digits of a given Integer.

Enter any integer as input. After this we perform various operations like modulus and division to extract each digit and increment it by one.

Here is the source code of the Java Program to Increment by 1 All the Digits of a given Integer. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Increment_Digits
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, m = 0, a;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any number:");
9.         n = s.nextInt();
10.        while(n > 0)
11.        {
12.            a = n % 10;
13.            a++;
14.            m = m * 10 + a;
15.            n = n / 10;
16.        }
17.        n = m;
18.        m = 0;
19.        while(n > 0)
20.        {
21.            a = n % 10;
22.            m = m * 10 + a;
23.            n = n / 10;
24.        }
25.        System.out.println("Result:"+m);
26.    }
27.}
```

Output:

```
$ javac Increment_Digits.java
$ java Increment_Digits
```

```
Enter any number:4567
Result:5678
```

40. Java Program to Illustrate Use of Binary Literals

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Binary Literals. A literal is an explicit value. To specify a binary literal, add the prefix 0b or 0B to the number. For example : int number=123; The value 123 is a literal. Since it's a normal base-10 value, we might say it is a decimal literal.

We use binary literals with different type of datatypes like byte, short and int to illustrate its use. We also use addition operation to add two binary literals.

Here is the source code of the Java Program to Illustrate Use of Binary Literals. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Binary_Literal
3. {
4.     public static void main(String[] args)
5.     {
6.         byte aB = 0b00100001;
7.         short aS = 0b10100010100;
8.         int a1 = 0b10110;
9.         int a2 = 0b101;
10.        int a3 = 0b1011;
11.        int aI=a2+a3;
12.        System.out.println("Byte value:"+aB);
13.        System.out.println("Short value:"+aS);
14.        System.out.println("Integer value:"+a1);
15.        System.out.println("Result:"+aI);
16.    }
17.}
```

Output:

```
$ javac Binary_Literal.java
$ java Binary_Literal
```

```
Byte value:33
Short value:1300
Integer value:22
Result:16
```

41. Java Program to Use Underscores in Numeric Literal

[« Prev](#)

[Next »](#)

This is a Java Program to Use Underscores in Numeric Literal. A literal is an explicit value. For example :

int number=123;

The value 123 is a literal. Since it's a normal base-10 value, we might say it is a decimal literal.

We have used underscore at various positions in different types of datatype values and identified which of them are valid and which are not.

Here is the source code of the Java Program to Use Underscores in Numeric Literal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Underscore_Numeric
3. {
4.     public static void main(String[] args)
5.     {
6.         // int a1 = _52;           Invalid; cannot put underscores at the start of a literal
7.         int a2 = 5_2;           // OK (decimal literal)
8.         System.out.println(a2);
9.         // int a3 = 52_;          Invalid; cannot put underscores at the end of a literal
10.        int a4 = 5_____2;      // OK (decimal literal)
11.        System.out.println(a4);
12.        // int a5 = 0_x52;        Invalid; cannot put underscores in the 0x radix prefix
13.        // int a6 = 0x_52;        Invalid; cannot put underscores at the beginning of a number
14.        int a7 = 0x5_2;          // OK (hexadecimal literal)
15.        System.out.println(a7);
16.        // int a8 = 0x52_;        Invalid; cannot put underscores at the end of a number
17.        int a9 = 0_52;          // OK (octal literal)
18.        System.out.println(a9);
19.        int a10 = 05_2;         // OK (octal literal)
20.        System.out.println(a10);
21.        // int a11 = 052_;        Invalid; cannot put underscores at the end of a number
22.        // float b1 = 3_.1415F;    Invalid; cannot put underscores adjacent to a decimal point
23.        // float b2 = 3._1415F;    Invalid; cannot put underscores adjacent to a decimal point
24.        float b3 = 3.1_415F;     // OK (float literal)
25.        System.out.println(b3);
26.        // long c1 = 9999_L;      Invalid; cannot put underscores prior to an L suffix
27.        long c2 = 99_99L;        // OK (long literal)
28.        System.out.println(c2);
29.    }
30.}
```

Output:

```
$ javac Underscore_Numeric.java
$ java Underscore_Numeric
```

```
52
52
82
42
42
3.1415
9999
```

42. Java Program to Shutdown or Turn Off the Computer in Linux

[« Prev](#)

[Next »](#)

This is a Java Program to Shutdown or Turn Off the Computer in Linux.

We first check the name of the operating system of the computer. If its Linux we proceed else we would not. Now we would give number of seconds as input after which we want to turn off computer and then concatenate it with the shutdown command to turn off the computer.

Here is the source code of the Java Program to Shutdown or Turn Off the Computer in Linux. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.io.IOException;
2. import java.util.Scanner;
3. public class Shutdown_System
4. {
5.     public static void main(String args[]) throws IOException
6.     {
7.         int sec;
8.         String operatingSystem = System.getProperty("os.name");
9.         System.out.println("Name of Operating System:" + operatingSystem);
10.        if(operatingSystem.equals("Linux"))
11.        {
12.            Runtime runtime = Runtime.getRuntime();
13.            Scanner s = new Scanner(System.in);
14.            System.out.print("Enter the no. of seconds after which you want your computer to shutdown:");
15.            sec = s.nextInt();
16.            Process proc = runtime.exec("shutdown -h -t " + sec);
17.            System.exit(0);
18.        }
19.        else
20.        {
21.            System.out.println("Unsupported operating system.");
22.        }
23.    }
24.}
```

Output:

```
$ javac Shutdown_System.java
$ java Shutdown_System
```

```
Name of Operating System:Linux
Enter the no. of seconds after which you want your computer to shutdown:5
```

43. Java Program to Illustrate Use of Relational Operators

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Relational Operators. The relational operators determine the relationship that one operand has to the other.

Enter any two integers as input. After that we use various relational operators on these given integers and hence get the outputs accordingly.

Here is the source code of the Java Program to Illustrate Use of Relational Operators. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Relational_Operators
3. {
4.     public static void main(String args[])
5.     {
6.         Scanner s= new Scanner(System.in);
7.         System.out.print("Enter first integer:");
8.         int a = s.nextInt();
9.         System.out.print("Enter second integer:");
10.        int b = s.nextInt();
11.        System.out.println("a == b : " + (a == b));
12.        System.out.println("a != b : " + (a != b));
13.        System.out.println("a > b : " + (a > b));
14.        System.out.println("a < b : " + (a < b));
15.        System.out.println("b >= a : " + (b >= a));
16.        System.out.println("b <= a : " + (b <= a));
17.    }
18.}
```

Output:

```
$ javac Relational_Operators.java
$ java Relational_Operators
```

```
Enter first integer:25
Enter second integer:30
a == b : false
a != b : true
a > b : false
a < b : true
b >= a : true
b <= a : false
```

44. Java Program to Illustrate Use of Pre and Post Increment and Decrement Operators

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Pre and Post Increment and Decrement Operators. The increment (++) operator increases its operand by one. The decrement (--) operator decreases its operand by one.

Enter an integer as input. After that we perform the pre and post increment and decrement operators on the given integer and hence get the respective output.

Here is the source code of the Java Program to Illustrate Use of Pre and Post Increment and Decrement Operators. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Increment_Decrement
3. {
4.     public static void main(String[] args)
5.     {
6.         int a, b, c, d, e;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any integer a:");
9.         a = s.nextInt();
10.        b = ++a;
11.        System.out.println("Result after Pre Increment a:"+a);
12.        System.out.println("Result after Pre Increment b:"+b);
13.        c = a++;
14.        System.out.println("Result after Pre Increment a:"+a);
15.        System.out.println("Result after Post Increment c:"+c);
16.        d = --a;
17.        System.out.println("Result after Pre Increment a:"+a);
18.        System.out.println("Result after Pre Decrement d:"+d);
19.        e = a--;
20.        System.out.println("Result after Pre Increment a:"+a);
21.        System.out.println("Result after Post Decrement e:"+e);
22.    }
23. }
```

Output:

```
$ javac Increment_Decrement.java
$ java Increment_Decrement
```

```
Enter any integer a:12
Result after Pre Increment a:13
Result after Pre Increment b:13
Result after Pre Increment a:14
Result after Post Increment c:13
Result after Pre Increment a:13
Result after Pre Decrement d:13
Result after Pre Increment a:12
Result after Post Decrement e:13
```

45. Java Program to Calculate Sum of Two Byte Values Using Type Casting

[« Prev](#)

[Next »](#)

This is a Java Program to Calculate Sum of Two Byte Values Using Type Casting. Type casting in Java is to cast one type, a class or interface, into another type i.e. another class or interface.

Enter any two byte values as input. After that we first convert these two bytes into integer using type casting and then we add that values and get the desired output.

Here is the source code of the Java Program to Calculate Sum of Two Byte Values Using Type Casting. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Byte_Sum
3. {
4.     public static void main(String[] args)
5.     {
6.         byte a, b;
7.         int x, y, z;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter first byte value:");
10.        a = s.nextByte();
11.        x = a;
12.        System.out.print("Enter second byte value:");
13.        b = s.nextByte();
14.        y = b;
15.        z = x + y;
16.        System.out.println("Result:"+z);
17.    }
18.}
```

Output:

```
$ javac Byte_Sum.java
$ java Byte_Sum
```

```
Enter first byte value:124
Enter second byte value:115
Result:239
```

46. Java Program to Perform String Concatenation

[« Prev](#)

[Next »](#)

This is a Java Program to Perform String Concatenation.

Enter any two strings as input. We have made a function into which we pass the two given strings and this function uses concatenation operator for string concatenation and hence return the desired string back to the main function.

Here is the source code of the Java Program to Perform String Concatenation. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class String_Concatenate
3. {
4.     public static void main(String[] args)
5.     {
6.         String a, b, c;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter first string:");
9.         a = s.nextLine();
10.        System.out.print("Enter second string:");
11.        b = s.nextLine();
12.        String_Concatenate obj = new String_Concatenate();
13.        c = obj.concat(a, b);
14.        System.out.println("New String:"+c);
15.    }
16.    String concat(String x, String y)
17.    {
18.        String z;
19.        z = x + " " + y;
20.        return z;
21.    }
22.}
```

Output:

```
$ javac String_Concatenate.java
$ java String_Concatenate
```

```
Enter first string:hello
Enter second string:world
New String:hello world
```

47. Java Program to Convert Long Values into Byte

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Long Values into Byte.

Enter any long integer as an input. After that we convert the given long integer into byte using explicit type casting.

Here is the source code of the Java Program to Convert Long Values into Byte. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Long_Byte
3. {
4.     public static void main(String[] args)
5.     {
6.         long a;
7.         byte b;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter any long value:");
10.        a = s.nextLong();
11.        b = (byte) a;
12.        System.out.println("Conversion into byte:"+b);
13.    }
14.}
```

Output:

```
$ javac Long_Byte.java
$ java Long_Byte
```

```
Enter any long value:12548
Conversion into byte:4
```

48. Java Program to Convert Integer Values into Byte, Character, Float

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Integer Values into Byte, Character, Float.

Enter any integer as input. After that we convert the given integer into byte and char by type casting (Explicit casting) but we can directly convert int value to float value as java itself performs this type of type casting (Implicit casting).

Here is the source code of the Java Program to Convert Integer Values into Byte, Character, Float. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Integer_Conversion
3. {
4.     public static void main(String[] args)
5.     {
6.         int a;
7.         byte b;
8.         char c;
9.         float d;
10.        Scanner s = new Scanner(System.in);
11.        System.out.print("Enter any integer:");
12.        a = s.nextInt();
13.        b = (byte) a;
14.        System.out.println("Conversion into byte:"+b);
15.        c = (char) a;
16.        System.out.println("Conversion into char:"+c);
17.        d = a;
18.        System.out.println("Conversion into float:"+d);
19.    }
20.}
```

Output:

```
$ javac Integer_Conversion.java
$ java Integer_Conversion
```

```
Enter any integer:97
Conversion into byte:97
Conversion into char:a
Conversion into float:97.0
```

49. Java Program to Illustrate Use of Final Keyword

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Final Keyword.

We have declared the length and breadth variable as final and hence its value cannot be changed now. We have made the area and rect method final so now these methods cannot be overridden. Also we have made the class containing main method as final so now this class cannot be inherited.

Here is the source code of the Java Program to Illustrate Use of Final Keyword. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Figure
3. {
4.     final int length = 5;
5.     final int breadth = 4;
6.     final void area()
7.     {
8.         int a = length * breadth;
9.         System.out.println("Area:"+a);
10.    }
11. }
12.class Rectangle extends Figure
13.
14. final void rect()
15. {
16.     System.out.println("This is rectangle");
17. }
18.
19.final public class Final_Use extends Rectangle
20.
21. public static void main(String[] args)
22. {
23.     Final_Use obj = new Final_Use();
24.     obj.rect();
25.     obj.area();
26. }
27.}
```

Output:

```
$ javac Final_Use.java
$ java Final_Use
```

```
This is rectangle
Area:20
```

50. Java Program to Calculate Sum & Average of an Array

[« Prev](#)

[Next »](#)

This is a Java Program to Calculate Sum & Average of an Array.

Enter size of array and then enter all the elements of that array. Now using for loop we calculate sum of elements of array and hence we divide it by number of elements in array to get average.

Here is the source code of the Java Program to Calculate Sum & Average of an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Sum_Average
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, sum = 0;
7.         float average;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter no. of elements you want in array:");
10.        n = s.nextInt();
11.        int a[] = new int[n];
12.        System.out.println("Enter all the elements:");
13.        for(int i = 0; i < n ; i++)
14.        {
15.            a[i] = s.nextInt();
16.            sum = sum + a[i];
17.        }
18.        System.out.println("Sum:"+sum);
19.        average = (float)sum / n;
20.        System.out.println("Average:"+average);
21.    }
22.}
```

Output:

```
$ javac Sum_Average.java
$ java Sum_Average
```

Enter no. of elements you want in array:5

Enter all the elements:

```
4
7
6
9
3
Sum:29
Average:5.8
```

51. Java Program to Delete the Specified Integer from an Array

[« Prev](#)

[Next »](#)

This is a Java Program to Delete the Specified Integer from an Array.

Enter size of array and then enter all the elements of that array. Now enter the element you want to delete. We first find the location of that element and then shift the positions of all the elements after the element to be removed by one.

Here is the source code of the Java Program to Delete the Specified Integer from an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Delete
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, x, flag = 1, loc = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for (int i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        System.out.print("Enter the element you want to delete:");
17.        x = s.nextInt();
18.        for (int i = 0; i < n; i++)
19.        {
20.            if(a[i] == x)
21.            {
22.                flag = 1;
23.                loc = i;
24.                break;
25.            }
26.            else
27.            {
28.                flag = 0;
29.            }
30.        }
31.        if(flag == 1)
32.        {
33.            for(int i = loc+1; i < n; i++)
34.            {
35.                a[i-1] = a[i];
36.            }
37.            System.out.print("After Deleting:");
38.            for (int i = 0; i < n-2; i++)
39.            {
40.                System.out.print(a[i]+",");
41.            }
42.            System.out.print(a[n-2]);
43.        }
44.        else
45.        {
46.            System.out.println("Element not found");
47.        }
48.    }}
```

Output:

```
$ javac Delete.java  
$ java Delete
```

Enter no. of elements you want in array:5

Enter all the elements:

```
3  
5  
8  
1  
4
```

Enter the element you want to delete:5

After Deleting:3,8,1,4

52. Java Program to Sort the Array in Descending Order

[« Prev](#)

[Next »](#)

This is a Java Program to Sort the Array in Descending Order.

Enter size of array and then enter all the elements of that array. Now with the help of for loop and temp variable we sort the array in descending order.

Here is the source code of the Java Program to Sort the Array in Descending Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Descending_Order
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, temp;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for (int i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        for (int i = 0; i < n; i++)
17.        {
18.            for (int j = i + 1; j < n; j++)
19.            {
20.                if (a[i] < a[j])
21.                {
22.                    temp = a[i];
23.                    a[i] = a[j];
24.                    a[j] = temp;
25.                }
26.            }
27.        }
28.        System.out.print("Descending Order:");
29.        for (int i = 0; i < n - 1; i++)
30.        {
31.            System.out.print(a[i] + ",");
32.        }
33.        System.out.print(a[n - 1]);
34.    }
35.}
```

Output:

```
$ javac Descending_Order.java
$ java Descending_Order
```

Enter no. of elements you want in array:5

Enter all the elements:

```
2
3
5
1
4
```

Descending Order:5,4,3,2,1

53. Java Program to Split an Array from Specified Position

[« Prev](#)

[Next »](#)

This is a Java Program to Split an Array from Specified Position.

Enter size of array and then enter all the elements of that array. Now enter the position from where you want to split. We first copy the elements from first position to that given position in secnd array and remaining elements in third array.

Here is the source code of the Java Program to Split an Array from Specified Position. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Split
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, x, flag = 1, loc = 0, k = 0, j = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        int b[] = new int[n];
12.        int c[] = new int[n];
13.        System.out.println("Enter all the elements:");
14.        for (int i = 0; i < n; i++)
15.        {
16.            a[i] = s.nextInt();
17.        }
18.        System.out.print("Enter the position from where you want to split:");
19.        loc = s.nextInt();
20.        for(int i = 0; i < loc; i++)
21.        {
22.            b[k] = a[i];
23.            k++;
24.        }
25.        for(int i = loc; i < n; i++)
26.        {
27.            c[j] = a[i];
28.            j++;
29.        }
30.        System.out.print("First array:");
31.        for(int i = 0; i < k; i++)
32.        {
33.            System.out.print(b[i] + " ");
34.        }
35.        System.out.println("");
36.        System.out.print("Second array:");
37.        for(int i = 0; i < j; i++)
38.        {
39.            System.out.print(c[i] + " ");
40.        }
41.    }
42.}
```

Output:

```
$ javac Split.java
$ java Split
```

Enter no. of elements you want in array:8

Enter all the elements:

2
3
4
7
1
9
11
6

Enter the position from where you want to split:4

First array:2 3 4 7

Second array:1 9 11 6

54. Java Program to Sort Names in an Alphabetical Order

[« Prev](#)

[Next »](#)

This is a Java Program to Sort Names in an Alphabetical Order.

Enter size of array and then enter all the names in that array. Now with the help of compareTo operator we can easily sort names in Alphabetical Order.

Here is the source code of the Java Program to Sort Names in an Alphabetical Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Alphabetical_Order
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         String temp;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter number of names you want to enter:");
10.        n = s.nextInt();
11.        String names[] = new String[n];
12.        Scanner s1 = new Scanner(System.in);
13.        System.out.println("Enter all the names:");
14.        for(int i = 0; i < n; i++)
15.        {
16.            names[i] = s1.nextLine();
17.        }
18.        for (int i = 0; i < n; i++)
19.        {
20.            for (int j = i + 1; j < n; j++)
21.            {
22.                if (names[i].compareTo(names[j])>0)
23.                {
24.                    temp = names[i];
25.                    names[i] = names[j];
26.                    names[j] = temp;
27.                }
28.            }
29.        }
30.        System.out.print("Names in Sorted Order:");
31.        for (int i = 0; i < n - 1; i++)
32.        {
33.            System.out.print(names[i] + ",");
34.        }
35.        System.out.print(names[n - 1]);
36.    }
37.}
```

Output:

```
$ javac Alphabetical_Order.java
$ java Alphabetical_Order
```

Enter number of names you want to enter:5

Enter all the names:

bryan
adam
rock
chris

scott

Names in Sorted Order:adam,bryan,chris,rock,scott

55. Java Program to Print the kth Element in the Array

[« Prev](#)

[Next »](#)

This is a Java Program to Print the kth Element in the Array.

Enter size of array and then enter all the elements of that array. Now enter the position k at which you want to find element. Now we print the element at k-1 position in given array.

Here is the source code of the Java Program to Print the kth Element in the Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Position
3. {
4.     public static void main(String[] args)
5.     {
6.         int n;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter no. of elements you want in array:");
9.         n = s.nextInt();
10.        int a[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for (int i = 0; i < n; i++)
13.        {
14.            a[i] = s.nextInt();
15.        }
16.        System.out.print("Enter the k th position at which you want to check number:");
17.        int k = s.nextInt();
18.        System.out.println("Number:" + a[k-1]);
19.    }
20.}
```

Output:

```
$ javac Position.java
$ java Position
```

Enter no. of elements you want in array:5

Enter all the elements:

```
2
5
3
8
6
```

Enter the k th position at which you want to check number:3

Number:3

56. Java Program to Find the Largest Two Numbers in a Given Array

[« Prev](#)

[Next »](#)

This is a Java Program to Find the Largest Two Numbers in a Given Array.

Enter size of array and then enter all the elements of that array. Now we first sort the array in decreasing order using double for loops and hence get the first two elements as output.

Here is the source code of the Java Program to Find the Largest Two Numbers in a Given Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class largest_and_second
3. {
4.     public static void main (String[] args)
5.     {
6.         Scanner scn = new Scanner (System.in);
7.         System.out.print("Enter no. of elements you want in array:");
8.         int n = scn.nextInt();
9.
10.        int array[] = new int[n];
11.        System.out.println("Enter all the elements:");
12.        for (int i = 0; i < array.length; i++)
13.        {
14.            array[i] = scn.nextInt();
15.        }
16.
17.        int largest1, largest2, temp;
18.
19.        largest1 = array[0];
20.        largest2 = array[1];
21.
22.        if (largest1 < largest2)
23.        {
24.            temp = largest1;
25.            largest1 = largest2;
26.            largest2 = temp;
27.        }
28.
29.        for (int i = 2; i < array.length; i++)
30.        {
31.            if (array[i] > largest1)
32.            {
33.                largest2 = largest1;
34.                largest1 = array[i];
35.            }
36.            else if (array[i] > largest2 && array[i] != largest1)
37.            {
38.                largest2 = array[i];
39.            }
40.        }
41.
42.        System.out.println ("The First largest is " + largest1);
43.        System.out.println ("The Second largest is " + largest2);
44.
45.    }
46. }
```

Output:

Enter no. of elements you want in array:5

Enter all the elements:

1
5
4
8
7

The First largest is 8

The Second largest is 7

Enter no. of elements you want in array:7

Enter all the elements:

3
2
1
3
2
1
3

The First largest is 3

The Second largest is 2

57. Java Program to Find the Number of Non-Repeated Elements in an Array

[« Prev](#)

[Next »](#)

This is the Java Program to Find the Elements that do Not have Duplicates.

Problem Description

Given an array, print all the elements whose frequency is one, that is they do not have duplicates.

advertisement

Example:

Array = [-1, -2, 3, 3, -2]

Output = -1

Problem Solution

Traverse through the array and **find the frequency of each element**, if the frequency is one, then print the element.

Program/Source Code

advertisement

Here is the source code of the Java Program to Find the Elements that do Not have Duplicates. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1.
2. //Java Program to Find the Elements that do Not have Duplicates
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5.
6. public class NoDuplicates {
7.     //Function to print elements with no duplicates
8.     static void printElementsWithNoDuplicates(int[] array) {
9.         int i,j;
10.        int count;
11.        int x = 0;
12.        boolean[] flag = new boolean[array.length];
13.        for(i=0; i<array.length; i++){
14.            if(!flag[i]){
15.                count = 1;
16.                for(j=i+1; j<array.length;j++){
17.                    if(array[j] == array[i])
18.                    {
19.                        count++;
20.                        flag[j] = true;
21.                    }
22.                }
23.                if(count == 1){
24.                    System.out.println(array[i]);
25.                    x++;
26.                }
27.            }
28.        }
29.        if(x==0){
30.            System.out.println("All elements are repeated");
31.        }
}
```

```

32. }
33. //Function to read input
34. public static void main(String[] args) {
35.     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
36.     int size;
37.     System.out.println("Enter the size of the array");
38.     try {
39.         size = Integer.parseInt(br.readLine());
40.     } catch (Exception e) {
41.         System.out.println("Invalid Input");
42.         return;
43.     }
44.     int[] array = new int[size];
45.     System.out.println("Enter array elements");
46.     int i;
47.     for (i = 0; i < array.length; i++) {
48.         try {
49.             array[i] = Integer.parseInt(br.readLine());
50.         } catch (Exception e) {
51.             System.out.println("An error Occurred");
52.         }
53.     }
54.     System.out.println("The elements are ");
55.     printElementsWithNoDuplicates(array);
56. }
57.

```

Program Explanation

1. In function `printElementsWithNoDuplicates()`, an array of boolean is created to keep track of the elements which are visited.
2. The loop `for(i=0; i<array.length; i++)` is used to iterate through the array.
3. The condition `if(!flag[i])` is used to check whether the element is not visited already.
4. If true, then the nested loop `for(j=i+1; j<array.length;j++)` is used to find the frequency of the element.
5. If the frequency is 1, then the element is printed.
6. If every element has a duplicate, then a suitable message is displayed.

advertisement

Time Complexity: $O(n^2)$ where n is the number of elements in the array.

Runtime Test Cases

Case 1 (Simple Test Case - some elements are unique):

Enter the size of the array

6

Enter array elements

1

2

3

4

5

The elements are

1

2

3

4

Case 2 (Simple Test Case - all elements have a duplicate):

Enter the size of the array

6

Enter array elements

1

2

3

3

2

1

The elements are

All elements are repeated

58. Java Program to Check whether a String is a Palindrome

[« Prev](#)

[Next »](#)

This is a Java Program to Check whether a String is a Palindrome.

Enter any string as input. Now we use for loops and if-else conditions along with equalsIgnoreCase() method to conclude whether the entered string is palindrome or not.

Here is the source code of the Java Program to Check whether a String is a Palindrome. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Palindrome
2. {
3.     public static void main(String args[])
4.     {
5.         String a, b = "";
6.         Scanner s = new Scanner(System.in);
7.         System.out.print("Enter the string you want to check:");
8.         a = s.nextLine();
9.         int n = a.length();
10.        for(int i = n - 1; i >= 0; i--)
11.        {
12.            b = b + a.charAt(i);
13.        }
14.        if(a.equalsIgnoreCase(b))
15.        {
16.            System.out.println("The string is palindrome.");
17.        }
18.        else
19.        {
20.            System.out.println("The string is not palindrome.");
21.        }
22.    }
23.}
```

Output:

```
$ javac Palindrome.java
$ java Palindrome
```

```
Enter the string you want to check:NeveroddorEVen
The string is palindrome.
```

59. Java Program to Illustrate Use of Constructor

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Constructor.

Here we have made two types of constructor. One is simple and other is parametric. Now depending upon the object created, constructor gets started and hence we get the output.

Here is the source code of the Java Program to Illustrate Use of Constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Constructor
2. {
3.     double width;
4.     double height;
5.     double depth;
6.     Constructor()
7.     {
8.         System.out.println("Constructor without parameter");
9.         width = 10;
10.        height = 10;
11.        depth = 10;
12.    }
13.    Constructor(int a, int b, int c)
14.    {
15.        System.out.println("Constructor with parameter");
16.        width = a;
17.        height = b;
18.        depth = c;
19.    }
20.    double volume()
21.    {
22.        return width * height * depth;
23.    }
24.}
25.class ConstructorDemo
26{
27.    public static void main(String args[])
28.    {
29.        Constructor cuboid1 = new Constructor();
30.        double vol;
31.        vol = cuboid1.volume();
32.        System.out.println("Volume is " + vol);
33.        Constructor cuboid2 = new Constructor(8,11,9);
34.        vol = cuboid2.volume();
35.        System.out.println("Volume is " + vol);
36.    }
37.}
```

Output:

```
$ javac ConstructorDemo.java
$ java ConstructorDemo
```

```
Constructor without parameter
Volume is 1000.0
Constructor with parameter
Volume is 792.0
```

60. Java Program to Illustrate Use of getChar() and split() Method

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of getChar() and split() Method.

Here we divide the given string into array of strings depending upon the positions of blank spaces. Also we use getChar() method to get characters from given given string.

Here is the source code of the Java Program to Illustrate Use of getChar() and split() Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Split_getChar
2. {
3.     public static void main(String args[])
4.     {
5.         String[] a;
6.         String b = "Sanfoundry 1000 Java Programs";
7.         a = b.split(" ");
8.         for (int i = 0; i < a.length; i++)
9.         {
10.             System.out.println(a[i]);
11.         }
12.         System.out.println("Using getChar() method:");
13.         char c[] = new char[11];
14.         b.getChars(0, 11, c, 0);
15.         for (int i = 0; i < 11; i++)
16.         {
17.             System.out.print(c[i]);
18.         }
19.     }
20. }
```

Output:

```
$ javac Split_getChar.java
$ java Split_getChar
```

```
Sanfoundry
1000
Java
Programs
Usin getChar() method:
Sanfoundry
```

61. Java Program to Illustrate Use of Methods in a Class

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of Methods in a Class.

Here we have made different methods to perform different arithamatic operations. Hence we call these methods from main method to get the output accordingly.

Here is the source code of the Java Program to Illustrate Use of Methods in a Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Method
2. {
3.     void addition(int a,int b)
4.     {
5.         int c = a + b;
6.         System.out.println("Result:"+c);
7.     }
8.     void subtraction(int a,int b)
9.     {
10.        int c = a - b;
11.        System.out.println("Result:"+c);
12.    }
13.    void multiplication(int a,int b)
14.    {
15.        int c = a * b;
16.        System.out.println("Result:"+c);
17.    }
18.    void division(int a,int b)
19.    {
20.        double c =(double)a / b;
21.        System.out.println("Result:"+c);
22.    }
23.    public static void main(String args[])
24.    {
25.        Method obj = new Method();
26.        obj.addition(10, 4);
27.        obj.subtraction(10, 4);
28.        obj.multiplication(10, 4);
29.        obj.division(10, 4);
30.    }
31.}
```

Output:

```
$ javac Method.java
$ java Method
```

```
Result:14
Result:6
Result:40
Result:2.5
```

62. Java Program to Illustrate a Method without Parameters and Without Return Type

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate a Method without Parameters and Without Return Type.

We have made the method to calculate area. We call this method from main method and then enter the radius of the circle as input. Now we calculate area and print the output.

Here is the source code of the Java Program to Illustrate a Method without Parameters and Without Return Type. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Test
2. {
3.     void areacircle()
4.     {
5.         System.out.print("enter the radius :");
6.         Scanner s = new Scanner(System.in);
7.         float r = s.nextFloat();
8.         float ar;
9.         ar = (r * r) * 22 / 7;
10.        System.out.println("area of the circle is : "+ar+" sq units.");
11.    }
12.}
13.class MethodDemo
14.
15. public static void main(String args[])
16. {
17.     Test obj = new Test();
18.     obj.areacircle();
19. }
20.}
```

Output:

```
$ javac Method_Demo.java
$ java Method_Demo
```

```
enter the radius :3
area of the circle is:28.285715 sq units.
```

63. Java Program to Illustrate a Method with 2 Parameters and Without Return Type

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate a Method with 2 Parameters and Without Return Type.

Enter the length and breadth of rectangle as input. Now we pass these values as parameters to the new method where we calculate the area and print the output.

Here is the source code of the Java Program to Illustrate a Method with 2 Parameters and Without Return Type. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
import java.util.Scanner;
public class Rectangle
{
    double width;
    double height;
    double depth;
    void recarea(double x,double y)
    {
        double ar = x * y;
        System.out.print("area of the rectangle is :" + ar);
    }
}
class ParameterDemo
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("enter length : ");
        double l = s.nextDouble();
        System.out.print("enter breadth : ");
        double b = s.nextDouble();
        Rectangle rec = new Rectangle();
        rec.recarea(l,b);
    }
}
```

Output:

```
$ javac ParameterDemo.java
$ java ParameterDemo
```

```
enter length : 3
enter breadth : 2
area of the rectangle is :6.0
```

64. Java Program to Test Whether a Static Method can Access the Static or Non-Static Variable

[« Prev](#)

[Next »](#)

This is a Java Program to Test whether a Static Method can Access the Static or Non-Static Variable.

Here we consider two variables, one as static and other as non static. Now when we print the value of these variables in a static method, we get only the value of the static variable and an exception is generated while trying to print non static variable.

Here is the source code of the Java Program to Test whether a Static Method can Access the Static or Non-Static Variable. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class StaticMethod
2. {
3.     static int a = 42;
4.     int b = 99;
5.     public static void main (String args[])
6.     {
7.         System.out.println("static variable:" + a);
8.         System.out.println("non static variable:" + b);
9.     }
10. }
```

Output:

```
$ javac StaticMethod.java
$ java StaticMethod

static variable:42
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - non-static variable
```

65. Java Program to Create the Outer Class BankAcct and Inner Class Interest

[« Prev](#)

[Next »](#)

This is a Java Program to Create the Outer Class BankAcct and Inner Class Interest.

We declare the Outer class as BankAcct and inner class as Interest. Now we use the variables declared in the outer class to get the output in the inner class.

Here is the source code of the Java Program to Create the Outer Class BankAcct and Inner Class Interest. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class BankAcct
2. {
3.     int principal = 200, rate = 4, time = 2;
4.     void test()
5.     {
6.         Interest inner_obj = new Interest();
7.         inner_obj.display();
8.     }
9.     class Interest
10.    {
11.        void display()
12.        {
13.            int si = (principal * rate * time) / 100;
14.            System.out.println("Interest : " + si + " Rs");
15.        }
16.    }
17.}
18.public class InnerClassDemo
19.
20.    public static void main(String args[])
21.    {
22.        BankAcct outer_obj = new BankAcct();
23.        outer_obj.test();
24.    }
25.}
```

Output:

```
$ javac InnerClassDemo.java
$ java InnerClassDemo
```

```
Interest : 16 Rs
```

66. Java Program to Illustrates Use of Instance Inner Class

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrates Use of Instance Inner Class. An inner class has access to all of the members of its enclosing class, but the reverse is not true.

Here we access the members of the outer class named outer in the inner class named inner. But the members of inner class cannot be used by the outer class.

Here is the source code of the Java Program to Illustrates Use of Instance Inner Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Outer
2. {
3.     int outer_x = 100;
4.     void test()
5.     {
6.         Inner inner = new Inner();
7.         inner.display();
8.     }
9.     class Inner
10.    {
11.        int y = 10;
12.        void display()
13.        {
14.            System.out.println("display: outer_x = " + outer_x);
15.        }
16.    }
17.    void showy()
18.    {
19.        System.out.println(y); // error, y not known here!
20.    }
21.}
22.class InnerDemo
23.{
24.    public static void main(String args[])
25.    {
26.        Outer outer = new Outer();
27.        outer.test();
28.    }
29.}
```

Output:

```
$ javac InnerDemo.java
$ java InnerDemo
```

```
display: outer_x = 100
```

67. Java Program to Print Diamond Pattern

[« Prev](#)

[Next »](#)

This is a Java Program to Print Diamond Pattern.

Enter the number of rows as an input. Now we use for loops to print two equilateral triangles facing away from each other but with same base.

Here is the source code of the Java Program to Print Diamond Pattern. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Diamond
3. {
4.     public static void main(String args[])
5.     {
6.         int n, i, j, space = 1;
7.         System.out.print("Enter the number of rows: ");
8.         Scanner s = new Scanner(System.in);
9.         n = s.nextInt();
10.        space = n - 1;
11.        for (j = 1; j <= n; j++)
12.        {
13.            for (i = 1; i <= space; i++)
14.            {
15.                System.out.print(" ");
16.            }
17.            space--;
18.            for (i = 1; i <= 2 * j - 1; i++)
19.            {
20.                System.out.print("*");
21.            }
22.            System.out.println("");
23.        }
24.        space = 1;
25.        for (j = 1; j <= n - 1; j++)
26.        {
27.            for (i = 1; i <= space; i++)
28.            {
29.                System.out.print(" ");
30.            }
31.            space++;
32.            for (i = 1; i <= 2 * (n - j) - 1; i++)
33.            {
34.                System.out.print("*");
35.            }
36.            System.out.println("");
37.        }
38.    }
39.}
```

Output:

```
$ javac Diamond.java
$ java Diamond
```

Enter the number of rows: 5

```
*  
***  
*****  
*****  
*****
```

*

68. Java Program to Find Sum of Natural Numbers Using While Loop

[« Prev](#)

[Next »](#)

This is a Java Program to Find Sum of Natural Numbers Using While Loop.

Enter the number upto which you want to calculate the sum. Now we use while loops till the given number to get the desired output.

Here is the source code of the Java Program to Find Sum of Natural Numbers Using While Loop. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Natural
2. {
3.     public static void main(String args[])
4.     {
5.         int x, i = 1 ;
6.         int sum = 0;
7.         System.out.println("Enter Number of items :");
8.         Scanner s = new Scanner(System.in);
9.         x = s.nextInt();
10.        while(i <= x)
11.        {
12.            sum = sum +i;
13.            i++;
14.        }
15.        System.out.println("Sum of "+x+" numbers is :" +sum);
16.    }
17.}
```

Output:

```
$ javac Natural.java
$ java Natural
```

```
Enter Number of items :
55
Sum of 55 numbers is :1540
```

69. Java Program to Find Prime Numbers Within a Range of n1 and n2

[« Prev](#)

[Next »](#)

This is a Java Program to Find Prime Numbers Within a Range of n1 and n2.

Enter the upper and lower limits as input. Now we use modulus operation along with double for loops and if-else conditions to get the output.

Here is the source code of the Java Program to Find Prime Numbers Within a Range of n1 and n2. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
import java.util.Scanner;
public class Prime
{
    public static void main(String args[])
    {
        int s1, s2, s3, flag = 0, i, j;
        Scanner s = new Scanner(System.in);
        System.out.println ("Enter the lower limit :");
        s1 = s.nextInt();
        System.out.println ("Enter the upper limit :");
        s2 = s.nextInt();
        System.out.println ("The prime numbers in between the entered limits are :");
        for(i = s1; i <= s2; i++)
        {
            for(j = 2; j < i; j++)
            {
                if(i % j == 0)
                {
                    flag = 0;
                    break;
                }
                else
                {
                    flag = 1;
                }
            }
            if(flag == 1)
            {
                System.out.println(i);
            }
        }
    }
}
```

Output:

```
$ javac Prime.java
$ java Prime
```

Enter the lower limit :

2

Enter the upper limit :

20

The prime numbers in between the entered limits are :

3

5

7

11

13

17

19

70. Java Program to Compute List of First 100 Fibonacci Numbers

[« Prev](#)

[Next »](#)

This is a Java Program to Compute List of First 100 Fibonacci Numbers. The number is said to be in a Fibonacci series if each subsequent number is the sum of the previous two numbers.

We use the loops assignment operator to get the desired output.

Here is the source code of the Java Program to Compute List of First 100 Fibonacci Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
import java.util.Scanner;
public class Fibonacci
{
    public static void main(String[] args)
    {
        int n, a = 0, b = 0, c = 1;
        System.out.print("Fibonacci Series:");
        for(int i = 1; i <= 100; i++)
        {
            a = b;
            b = c;
            c = a + b;
            System.out.print(a+" ");
        }
    }
}
```

Output:

```
$ javac Fibonacci.java
$ java Fibonacci
```

```
Fibonacci Series:0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657
```

71. Java Program to Convert a Decimal Number to Binary & Count the Number of 1s

[« Prev](#)

[Next »](#)

This is a Java Program to Convert a Decimal Number to Binary & Count the Number of 1s.

Enter any integer as an input. Now we convert the given decimal input into a binary number with the help of division and modulus operations along with loops and if conditions to get the desired output.

Here is the source code of the Java Program to Convert a Decimal Number to Binary & Count the Number of 1s. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Convert
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, count = 0, a;
7.         String x = "";
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter any decimal number:");
10.        n = s.nextInt();
11.        while(n > 0)
12.        {
13.            a = n % 2;
14.            if(a == 1)
15.            {
16.                count++;
17.            }
18.            x = a + "" + x;
19.            n = n / 2;
20.        }
21.        System.out.println("Binary number:"+x);
22.        System.out.println("No. of 1s:"+count);
23.    }
24.}
```

Output:

```
$ javac Convert.java
$ java Convert
```

```
Enter any decimal number:25
Binary number:11001
No. of 1s:3
```

72. Java Program to Find if a Number is Prime or Not using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Find if a Number is Prime or Not using Recursion. A number is said to be a prime number if it is divisible only by itself and unity.

Enter an integer as an input. Now we create a new method named prime which uses if conditons to give the desired result.

Here is the source code of the Java Program to Find if a Number is Prime or Not using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1.
2. import java.util.Scanner;
3. public class Prime
4. {
5.     public static void main(String[] args)
6.     {
7.         int n, x;
8.         Scanner s = new Scanner(System.in);
9.         System.out.print("Enter any number:");
10.        n = s.nextInt();
11.        Prime obj = new Prime();
12.        x = obj.prime(n, 2);
13.        if(x == 1)
14.        {
15.            System.out.println(n+" is prime number");
16.        }
17.        else
18.        {
19.            System.out.println(n+" is not prime number");
20.        }
21.    }
22.    int prime(int y,int i)
23.    {
24.        if(i < y)
25.        {
26.            if(y % i != 0)
27.            {
28.                return(prime(y, ++i));
29.            }
30.            else
31.            {
32.                return 0;
33.            }
34.        }
35.        return 1;
36.    }
37. }
```

Output:

```
$ javac Prime.java
$ java Prime
```

```
Enter any number:17
17 is prime number
```

73. Java Program to Find Factorial Value With Using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Find Factorial Value With Using Recursion.

Enter any integer of which you want to find factorial as an input. Now we make a new method named fact which calls itself until we get the final output.

Here is the source code of the Java Program to Find Factorial Value With Using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Factorial
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, mul;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any integer:");
9.         n = s.nextInt();
10.        Factorial obj = new Factorial();
11.        mul = obj.fact(n);
12.        System.out.println("Factorial of "+n+" :" +mul);
13.    }
14.    int fact(int x)
15.    {
16.        if(x > 1)
17.        {
18.            return(x * fact(x - 1));
19.        }
20.        return 1;
21.    }
22.}
```

Output:

```
$ javac Factorial.java
$ java Factorial
```

```
Enter any integer:8
Factorial of 8 :40320
```

74. Java Program to Find Factorial Value Without Using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to Find Factorial Value Without Using Recursion.

Enter any integer of which you want to find factorial as an input. Now we use for loop to calculate the factorial of given number.

Here is the source code of the Java Program to Find Factorial Value Without Using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Factorial
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, mul = 1;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any integer:");
9.         n = s.nextInt();
10.        for(int i = 1; i <= n; i++)
11.        {
12.            mul = mul * i;
13.        }
14.        System.out.println("Factorial of "+n+" :" +mul);
15.    }
16.}
```

Output:

```
$ javac Factorial.java
$ java Factorial
```

```
Enter any integer:8
Factorial of 8 :40320
```

75. Java Program to Print Armstrong Number from 1 to 1000

[« Prev](#)

[Next »](#)

This is a Java Program to Print Armstrong Number from 1 to 1000. Armstrong Number is an integer such that the sum of the cubes of its digits is equal to the number itself.

We use the Modulo and division operation along with loops and if conditions to get the desired output.

Here is the source code of the Java Program to Print Armstrong Number from 1 to 1000. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Armstrong
2. {
3.     public static void main(String[] args)
4.     {
5.         int n, count = 0, a, b, c, sum = 0;
6.         System.out.print("Armstrong numbers from 1 to 1000:");
7.         for(int i = 1; i <= 1000; i++)
8.         {
9.             n = i;
10.            while(n > 0)
11.            {
12.                b = n % 10;
13.                sum = sum + (b * b * b);
14.                n = n / 10;
15.            }
16.            if(sum == i)
17.            {
18.                System.out.print(i+" ");
19.            }
20.            sum = 0;
21.        }
22.    }
23.}
```

Output:

```
$ javac Armstrong.java
$ java Armstrong
```

```
Armstrong numbers from 1 to 1000:1 153 370 371 407
```

76. Java Program to Create a Simple Class to Find out the Area and Perimeter of Rectangle

« Prev

Next »

This is a Java Program to Create a Simple Class to Find out the Area and Perimeter of Rectangle. Formula:

Perimeter of rectangle = $2 * (\text{length} + \text{breadth})$

Area of rectangle = $\text{length} * \text{breadth}$

Enter the length and breadth of the rectangle as input. Now we use the given formula to calculate the area and perimeter of rectangle.

Here is the source code of the Java Program to Create a Simple Class to Find out the Area and Perimeter of Rectangle. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Area_Perimeter
3. {
4.     public static void main(String[] args)
5.     {
6.         int l, b, perimeter, area;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter length of rectangle:");
9.         l = s.nextInt();
10.        System.out.print("Enter breadth of rectangle:");
11.        b = s.nextInt();
12.        perimeter = 2 * (l + b);
13.        System.out.println("Perimeter of rectangle:" +perimeter);
14.        area = l * b;
15.        System.out.println("Area of rectangle:" +area);
16.    }
17. }
```

Output:

```
$ javac Area_Perimeter.java
$ java Area_Perimeter
```

```
Enter length of rectangle:4
Enter breadth of rectangle:5
Perimeter of rectangle:18
Area of rectangle:20
```

77. Java Program to Check If a Given Number is Armstrong Number

« Prev

Next »

This is a Java Program to Check If a Given Number is Armstrong Number. Armstrong Number is an integer such that the sum of the cubes of its digits is equal to the number itself

Enter the integer you want to check as an input. Now we use modulus and division operations along with loops and if else statements to get the output.

Here is the source code of the Java Program to Check If a Given Number is Armstrong Number. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Armstrong
3. {
4.     public static void main(String[] args)
5.     {
6.         int n, count = 0, a, b, c, sum = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter any integer you want to check:");
9.         n = s.nextInt();
10.        a = n;
11.        c = n;
12.        while(a > 0)
13.        {
14.            a = a / 10;
15.            count++;
16.        }
17.        while(n > 0)
18.        {
19.            b = n % 10;
20.            sum = (int) (sum+Math.pow(b, count));
21.            n = n / 10;
22.        }
23.        if(sum == c)
24.        {
25.            System.out.println("Given number is Armstrong");
26.        }
27.        else
28.        {
29.            System.out.println("Given number is not Armstrong");
30.        }
31.    }
32.}
```

Output:

```
$ javac Armstrong.java
$ java Armstrong
```

```
Enter any integer you want to check:153
Given number is Armstrong
```

78. Java Program to Find the Area of Parallelogram

[« Prev](#)

[Next »](#)

This is the Java Program to Find the Area of a Parallelogram.

Problem Description

Given the dimensions of a parallelogram, find out its area.

Problem Solution

The area of a parallelogram can be calculated using the formula:

advertisement

Area = base*height.

Program/Source Code

Here is the source code of the Java Program to Find the Area of a Parallelogram. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1. 
2. //Java Program to Find the Area of a Parallelogram
3. 
4. import java.io.BufferedReader;
5. import java.io.InputStreamReader;
6. 
7. public class AreaOfAParallelogram {
8.     // Function to calculate the area of a parallelogram
9.     public static void main(String[] args) {
10.         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11.         double base,height;
12.         System.out.println("Enter the base and the height of the parallelogram");
13.         try{
14.             base=Double.parseDouble(br.readLine());
15.             height=Double.parseDouble(br.readLine());
16.         }catch (Exception e){
17.             System.out.println("An error occurred");
18.             return;
19.         }
20.         if(base<=0 || height<=0){
21.             System.out.println("Wrong Input");
22.             return;
23.         }
24.         System.out.println("Area = " + base*height );
25.     }
26. }
```

Program Explanation

1. In function main(), first, the base and height of the parallelogram are taken as input.
2. The condition if(base<=0 || height<=0) checks if the input is valid or not.
3. The print statement displays the area of the parallelogram.

advertisement

Time Complexity: O(1)

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the base and the height of the parallelogram

8.34

5.678

Area = 47.35452

79. Java Program to Convert Fahrenheit into Celsius

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Fahrenheit into Celsius.
formula:

temperature in celsius = (temperature in fahrenheit - 32)*(0.5556)

Enter the temperature in Fahrenheit and now using the mentioned formula we get the desired output.

Here is the source code of the Java Program to Convert Fahrenheit into Celsius. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. import java.util.Scanner;
2. public class Fahrenheit_Celsius
3. {
4.     public static void main(String[] args)
5.     {
6.         double celsius, fahrenheit;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter temperature in Fahrenheit:");
9.         fahrenheit = s.nextDouble();
10.        celsius = (fahrenheit-32)*(0.5556);
11.        System.out.println("Temperature in Celsius:"+celsius);
12.    }
13.}
```

Output:

advertisement

```
$ javac Fahrenheit_Celsius.java
$ java Fahrenheit_Celsius
```

```
Enter temperature in Fahrenheit:15
Temperature in Celsius:-9.4452
```

80. Java Program to Accept a Matrix of Order MxN & Interchange the Diagonals

[« Prev](#)

[Next »](#)

This is a Java Program to Accept a Matrix of Order MxN & Interchange the Diagonals.

Enter the elements of array as input. Now we use loops and if else condition to interchange the two diagonals of the matrix.

Here is the source code of the Java Program to Accept a Matrix of Order MxN & Interchange the Diagonals. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Interchange_Diagonals
3. {
4.     public static void main(String[] args)
5.     {
6.         int p, q, temp = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter number of rows in matrix:");
9.         p = s.nextInt();
10.        System.out.print("Enter number of columns in matrix:");
11.        q = s.nextInt();
12.        if (p == q)
13.        {
14.            int a[][] = new int[p][q];
15.            System.out.println("Enter all the elements of matrix:");
16.            for (int i = 0; i < p; i++)
17.            {
18.                for (int j = 0; j < q; j++)
19.                {
20.                    a[i][j] = s.nextInt();
21.                }
22.            }
23.            System.out.println("Given Matrix:");
24.            for (int i = 0; i < p; i++)
25.            {
26.                for (int j = 0; j < q; j++)
27.                {
28.                    System.out.print(a[i][j] + " ");
29.                }
30.                System.out.println("");
31.            }
32.            for(int j = 0; j < q; j++)
33.            {
34.                temp = a[j][j];
35.                a[j][j] = a[j][q-1-j];
36.                a[j][q-1-j] = temp;
37.            }
38.            System.out.println("Matrix after interchanging diagonals");
39.            for (int i = 0; i < p; i++)
40.            {
41.                for (int j = 0; j < q; j++)
42.                {
43.                    System.out.print(a[i][j] + " ");
44.                }
45.                System.out.println("");
46.            }
47.        }
48.        else
49.    }
```

```
50.     System.out.println("Rows not equal to column");
51. }
52. }
53.}
```

Output:

```
$ javac Interchange_Diagonals.java
$ java Interchange_Diagonals
```

```
Enter number of rows in matrix:3
Enter number of columns in matrix:3
Enter all the elements of matrix:
```

```
1
2
3
4
5
6
7
8
9
```

Given Matrix:

```
1 2 3
4 5 6
7 8 9
```

Matrix after interchanging diagonals

```
3 2 1
4 5 6
9 8 7
```

81. Java Program to Interchange any two Rows & Columns in the given Matrix

[« Prev](#)

[Next »](#)

This is a Java Program to Interchange any two Rows & Columns in the given Matrix.

Enter the elements of array as input. Now select the option whether you want to interchange rows or columns. We use loops to interchange columns or rows respectively.

Here is the source code of the Java Program to Interchange any two Rows & Columns in the given Matrix. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Interchange
3. {
4.     public static void main(String[] args)
5.     {
6.         int p, q, n, x , y, temp = 0, k = 0;
7.         Scanner s = new Scanner(System.in);
8.         System.out.print("Enter number of rows in matrix:");
9.         p = s.nextInt();
10.        System.out.print("Enter number of columns in matrix:");
11.        q = s.nextInt();
12.        int a[][] = new int[p][q];
13.        System.out.println("Enter all the elements of matrix:");
14.        for (int i = 0; i < p; i++)
15.        {
16.            for (int j = 0; j < q; j++)
17.            {
18.                a[i][j] = s.nextInt();
19.            }
20.        }
21.        System.out.println("Given Matrix:");
22.        for (int i = 0; i < p; i++)
23.        {
24.            for (int j = 0; j < q; j++)
25.            {
26.                System.out.print(a[i][j] + " ");
27.            }
28.            System.out.println("");
29.        }
30.        while (true)
31.        {
32.            System.out.println("Enter 1 to interchange rows");
33.            System.out.println("Enter 2 to interchange columns");
34.            System.out.println("Enter 3 to Exit");
35.            n=s.nextInt();
36.            switch (n)
37.            {
38.                case 1:
39.                    System.out.println("Enter the row numbers:");
40.                    x = s.nextInt();
41.                    y = s.nextInt();
42.                    for(int i = 0; i < p; i++)
43.                    {
44.                        temp = a[(x-1)][i];
45.                        a[x-1][i] = a[y-1][i];
46.                        a[y-1][i] = temp;
47.                    }
48.                    System.out.println("Matrix after interchanging rows:"+x +" and "+y);
49.                    for (int i = 0; i < p; i++)
```

```

50.      {
51.          for (int j = 0; j < q; j++)
52.          {
53.              System.out.print(a[i][j] + " ");
54.          }
55.          System.out.println("");
56.      }
57.      break;
58.  case 2:
59.      System.out.println("Enter the column numbers:");
60.      x = s.nextInt();
61.      y = s.nextInt();
62.      for(int i = 0; i < p; i++)
63.      {
64.          temp = a[i][(x-1)];
65.          a[i][x-1] = a[i][(y-1)];
66.          a[i][y-1] = temp;
67.      }
68.      System.out.println("Matrix after interchanging columns:"+x +" and "+y);
69.      for (int i = 0; i < p; i++)
70.      {
71.          for (int j = 0; j < q; j++)
72.          {
73.              System.out.print(a[i][j] + " ");
74.          }
75.          System.out.println("");
76.      }
77.      break;
78.  case 3:
79.      System.exit(0);
80.  }
81. }
82. }
83. }
```

Output:

```
$ javac Interchange.java
$ java Interchange
```

```
Enter number of rows in matrix:3
Enter number of columns in matrix:3
Enter all the elements of matrix:
```

```
1
2
3
4
5
6
7
8
9
```

Given Matrix:

```
1 2 3
4 5 6
7 8 9
```

Enter 1 to interchange rows

Enter 2 to interchange columns

Enter 3 to Exit

```
1
```

Enter the row numbers:

```
2
3
```

Matrix after interchanging rows:2 and 3

```
1 2 3
7 8 9
```

4 5 6

Enter 1 to interchange rows

Enter 2 to interchange columns

Enter 3 to Exit

2

Enter the column numbers:

1

2

Matrix after interchanging columns:1 and 2

2 1 3

8 7 9

5 4 6

Enter 1 to interchange rows

Enter 2 to interchange columns

Enter 3 to Exit

3

82. Java Program to Display Upper/Lower Triangle of a Matrix

[« Prev](#)

[Next »](#)

This is the Java Program to Display Upper/Lower Triangle of a Matrix.

Problem Description

Given a square matrix, print it's upper and lower triangle.

Example:

Matrix:

```
0 0 1 1  
0 1 1 1  
0 0 0 1  
0 0 0 0
```

Output:

Upper Triangle

```
0 0 1 1  
0 1 1  
0 0  
0
```

advertisement

Lower Triangle

```
1  
0 1  
0 0 0
```

Problem Solution

Traverse through the matrix. To display the upper triangle **check whether the sum of i and j indexes is less than the order of the matrix** if it is then print the element `matrix[i][j]`. Similarly, to display the lower triangle **check whether the sum of i and j indexes is greater than or equal to the order of the matrix** if it is then print the element `matrix[i][j]`.

Program/Source Code

Here is the source code of the Java Program to Display Upper/Lower Triangle of a Matrix. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

advertisement

```
1.  
2. //Java Program to Display Upper/Lower Triangle of a Matrix  
3.  
4. import java.io.BufferedReader;  
5. import java.io.InputStreamReader;  
6.  
7. public class UpperAndLowerTriangle {  
8.     // Function to display upper and lower triangle  
9.     static void displayUpperAndLowerTriangle(int[][] matrix){  
10.        int order = matrix.length;  
11.        int i,j;  
12.        for(i=0; i<order; i++){  
13.            for(j=0; j<order; j++){  
14.                if((i+j) < order)
```

```

15.         System.out.print(matrix[i][j] + "\t");
16.     }
17.     System.out.println();
18. }
19. for(i=0; i<order; i++){
20.     for(j=0; j<order; j++){
21.         if((i+j) >= order)
22.             System.out.print(matrix[i][j] + "\t");
23.     }
24.     System.out.println();
25. }
26. }
27. // Function to read user input
28. public static void main(String[] args) {
29.     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
30.     int order;
31.     System.out.println("Enter the order of the matrix");
32.     try{
33.         order = Integer.parseInt(br.readLine());
34.     }
35.     catch(Exception e){
36.         System.out.println("An error occurred");
37.         return;
38.     }
39.     int[][] matrix = new int[order][order];
40.     System.out.println("Enter matrix elements");
41.     int i,j;
42.     for(i=0; i<order; i++){
43.         for(j=0; j<order; j++){
44.             try{
45.                 matrix[i][j] = Integer.parseInt(br.readLine());
46.             }
47.             catch(Exception e){
48.                 System.out.println("An error occurred");
49.                 return;
50.             }
51.         }
52.     }
53.     System.out.println("The matrix is");
54.     for(i=0; i<order; i++){
55.         for(j=0; j<order; j++){
56.             System.out.print(matrix[i][j] + "\t");
57.         }
58.         System.out.println();
59.     }
60.     System.out.println("The upper and lower triangle is");
61.     displayUpperAndLowerTriangle(matrix);
62. }
63. }

```

Program Explanation

1. In function `displayUpperAndLowerTriangle()`, the `order` variable is initialized to `matrix.length`, which is the order of the matrix passed.
2. Within the first set of nested loops, the condition `if((i+j) < order)` is checked to display the upper triangle of the matrix.
3. Within the second set of nested loops, the condition `if((i+j) >= order)` is checked to display the lower triangle of the matrix.

Time Complexity: $O(nm)$ where n is the number of rows and m is the number of columns in the matrix respectively.

advertisement

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the order of the matrix

4

Enter matrix elements

0

0

1

1

0

1

1

0

0

0

1

0

0

0

0

The matrix is

0	0	1	1
0	1	1	1
0	0	0	1
0	0	0	0

The upper and lower triangle is

0	0	1	1
0	1	1	
0	0		
0			

		1	
	0	1	
0	0	0	

Case 2 (Simple Test Case - another example):

Enter the order of the matrix

3

Enter matrix elements

1

2

3

4

5

6

7

8

9

The matrix is

1	2	3
4	5	6
7	8	9

The upper and lower triangle is

1	2	3
4	5	
7		

	6	
8	9	

84. Java Program to Find the Trace & Normal of a given Matrix

[« Prev](#)

[Next »](#)

This is a Java Program to Find the Trace & Normal of a given Matrix. The Trace of a matrix is the sum of main diagonal whereas the Normal is the square root of the sum of squares of all elements of a matrix.

Enter all the elements of matrix as input. We find the trace and normal of matrix using loops.

Here is the source code of the Java Program to Find the Trace & Normal of a given Matrix. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.*;
2. public class Trace
3. {
4.     public static void main(String args[])
5.     {
6.         int array[][]=new int[10][10];
7.         int i,j;
8.         double sum = 0, square = 0, result = 0;
9.         System.out.println("Enter total rows and columns: ");
10.        Scanner s = new Scanner(System.in);
11.        int row = s.nextInt();
12.        int column = s.nextInt();
13.        System.out.println("Enter matrix:");
14.        for(i = 0; i < row; i++)
15.        {
16.            for(j = 0; j < column; j++)
17.            {
18.                array[i][j] = s.nextInt();
19.                System.out.print(" ");
20.            }
21.        }
22.        System.out.println("The entered Matrix is :");
23.        for(i = 0; i < row; i++)
24.        {
25.            for(j = 0; j < column; j++)
26.            {
27.                System.out.print(array[i][j]+" ");
28.            }
29.            System.out.println(" ");
30.        }
31.        System.out.println("The Trace of the above matrix is ");
32.        for(i = 0; i < row; i++)
33.        {
34.            for(j = 0; j < column; j++)
35.            {
36.                if(i == j)
37.                {
38.                    sum = sum + (array[i][j]);
39.                }
40.            }
41.        }
42.        System.out.println(sum);
43.        System.out.println("The Normal of the above matrix is ");
44.        for(i = 0; i < row; i++)
45.        {
46.            for(j = 0; j < column; j++)
47.            {
48.                square = square + (array[i][j])*(array[i][j]);
49.            }
}
```

```
50.        }
51.    result = Math.sqrt(square);
52.    System.out.println(result);
53. }
54.}
```

Output:

```
$ javac Trace.java
$ java Trace
```

Enter total rows and columns:

3

3

Enter matrix:

1

2

3

4

5

6

7

8

9

The entered Matrix is :

1 2 3

4 5 6

7 8 9

The Trace of the above matrix is

15.0

The Normal of the above matrix is

16.881943016134134

85. Java Program to Find the Area of Rhombus

« Prev

Next »

This is the Java Program to Find the Area of a Rhombus.

Problem Description

Given the dimensions of a rhombus, find out its area.

Problem Solution

The area of a rhombus can be calculated using the formula:

advertisement

$$\text{Area} = (\text{diagonal1}) * (\text{diagonal2}) / 2.$$

Program/Source Code

Here is the source code of the Java Program to Find the Area of a Rhombus. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1.
2. //Java Program to Find the Area of a Rhombus
3.
4. import java.io.BufferedReader;
5. import java.io.InputStreamReader;
6.
7. public class AreaOfARhombus {
8.     // Function to calculate the area of a rhombus
9.     public static void main(String[] args) {
10.         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11.         double l1,l2;
12.         System.out.println("Enter the length of the diagonals of the rhombus");
13.         try{
14.             l1=Double.parseDouble(br.readLine());
15.             l2=Double.parseDouble(br.readLine());
16.         }catch (Exception e){
17.             System.out.println("An error occurred");
18.             return;
19.         }
20.         if(l1<=0 || l2<=0){
21.             System.out.println("Wrong input");
22.             return;
23.         }
24.         System.out.println("Area = " + (l1*l2)/2 );
25.     }
26. }
```

Program Explanation

1. In function main(), first, the length of the diagonals of the rhombus is taken as input.
2. The condition if($l1 \leq 0 \ || \ l2 \leq 0$) checks if the input is valid or not.
3. The print statement displays the area of the rhombus.

advertisement

Time Complexity: O(1)

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the length of the diagonals of the rhombus

23

45

Area = 517.5

86. Java Program to Find the Area of Trapezium

« Prev

Next »

This is the Java Program to Find the Area of a Trapezium.

Problem Description

Given the dimensions of a trapezium, find out its area.

Problem Solution

The area of a trapezium can be calculated using the formula:

advertisement

$$\text{Area} = (\text{sum of parallel sides}) * (\text{height}) / 2.$$

Program/Source Code

Here is the source code of the Java Program to Find the Area of a Trapezium. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1.
2. //Java Program to Find the Area of a Trapezium
3.
4. import java.io.BufferedReader;
5. import java.io.InputStreamReader;
6.
7. public class AreaOfATrapezium {
8.     // Function to find area of a trapezium
9.     public static void main(String[] args) {
10.         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11.         double l1,l2,height;
12.         System.out.println("Enter the length of the two parallel sides
13.                         and the height of the trapezium");
14.         try{
15.             l1=Double.parseDouble(br.readLine());
16.             l2=Double.parseDouble(br.readLine());
17.             height=Double.parseDouble(br.readLine());
18.         }catch (Exception e){
19.             System.out.println("An error occurred");
20.             return;
21.         }
22.         if(l1<=0 || l2<=0 || height<=0){
23.             System.out.println("Wrong Input");
24.         }
25.         System.out.println("Area = " + (l1+l2)*height/2 );
26.     }
27. }
```

Program Explanation

1. In function main(), first, the length of parallel sides and height of the trapezium is taken as input.
2. The condition if(l1<=0 || l2<=0 || height<=0) checks if the input is valid or not.
3. The print statement displays the area of the trapezium.

advertisement

Time Complexity: O(1)

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the length of the two parallel sides and the height of the trapezium

13.765

23.555

8.0

Area = 149.28

87. Java Program to Illustrates Use of Referencing the Object from Inner Class

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrates Use of Referencing the Object from Inner Class.

It is possible to define a class within another class, such classes are known as nested classes. The most important type of nested class is the inner class. An inner class is a non-static nested class. It has access to all of the variables and methods of its outer class and may refer to them directly in the same way that other non-static members of the outer class do.

Here is the source code of the Java Program to Illustrates Use of Referencing the Object from Inner Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Outer_Class
2. {
3.     static Outer_Class.InnerClass obj;
4.     void test1()
5.     {
6.         System.out.println("Success");
7.     }
8.     static public class InnerClass
9.     {
10.         private String name = "Peakit";
11.         public void test2()
12.         {
13.             Outer_Class outer = new Outer_Class();
14.             outer.test1();
15.         }
16.     }
17.     public static void main(String[] args)
18.     {
19.         obj = new Outer_Class.InnerClass();
20.         obj.test2();
21.     }
22. }
```

Output:

```
$ javac Outer_Class.java
$ java Outer_Class
```

```
Success
```

88. Java Program to Access the Super Class Method and Instance Variable Without Using Super Keyword from Sub Class

[« Prev](#)

[Next »](#)

This is a Java Program to Access the Super Class Method and Instance Variable Without Using Super Keyword from Sub Class.

We inherit the class Super in class Sub using the concept of Inheritance. Due to which all the Methods and Instance Variable of class Super (which are not private)are available to class Sub and we can directly use the Methods and Instance Variable without using Super Keyword.

Here is the source code of the Java Program to Access the Super Class Method and Instance Variable Without Using Super Keyword from Sub Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Super
2. {
3.     int a = 5, b = 6;
4.     int add()
5.     {
6.         int c = a + b;
7.         return c;
8.     }
9. }
10. public class Sub extends Super
11. {
12.     void show()
13.     {
14.         System.out.println("a:"+a);
15.         System.out.println("b:"+b);
16.         System.out.println("Result:"+add());
17.     }
18.     public static void main(String[] args)
19.     {
20.         Sub obj = new Sub();
21.         obj.show();
22.     }
23. }
```

Output:

```
$ javac Sub.java
$ java Sub
```

```
a:5
b:6
Result:11
```

89. Java Program to Access the Super Class Method and Instance Variable Using Super Keyword from Sub Class

[« Prev](#)

[Next »](#)

This is a Java Program to Access the Super Class Method and Instance Variable Using Super Keyword from Sub Class.

Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super. The second

is

used to access a member of the superclass that has been hidden by a member of a subclass.

Here is the source code of the Java Program to Access the Super Class Method and Instance Variable Using Super Keyword from Sub Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Parent
2. {
3.     int x = 5;
4.     void show()
5.     {
6.         System.out.println("Method of parent class using Super Keyword");
7.     }
8. }
9. class Child extends Parent
10.{
11.    int x = 9;
12.    void show()
13.    {
14.        System.out.println("Instance Variable of Parent class using Super Keyword :" +super.x);
15.        System.out.println("Instance variable of Child class :" +x);
16.        super.show();
17.        System.out.println("Method of Child class ");
18.    }
19.    public static void main(String... a)
20.    {
21.        Child ob = new Child();
22.        ob.show();
23.    }
24.}
```

Output:

```
$ javac Child.java
$ java Child
```

```
Instance Variable of Parent class using Super Keyword :5
Instance variable of Child class :9
Method of parent class using Super Keyword
Method of Child class
```

90. Java Program to Prove that the Default Constructor of the Super Class is Available to Sub Class by Default

[« Prev](#)

[Next »](#)

This is a Java Program to Prove that the Default Constructor of the Super Class is Available to Sub Class by Default.

Whenever a child class constructor is executed it has to call immediate parent class constructor first then itself. If a child class constructor is failed to call immediate parent class constructor first then it cannot run itself. Thus it gives us a proof that Default Constructor of the Super Class is Available to Sub Class by Default.

Here is the source code of the Java Program to Prove that the Default Constructor of the Super Class is Available to Sub Class by Default. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Sup
2. {
3.     static int a, b;
4.     Sup()
5.     {
6.         System.out.println("super class default constructor");
7.     }
8. }
9. public class Sub extends Sup
10.
11. Sub()
12. {
13.     System.out.println("sub class default constructor");
14. }
15. public static void main(String[] args)
16.
17.     new Sub();
18. }
19.
```

Output:

```
$ javac Sub.java
$ java Sub

super class default constructor
sub class default constructor
```

91. Java Program to Understand that the Parameterized Constructor of the Super Class can be called from Sub Class Using super()

[« Prev](#)

[Next »](#)

This is a Java Program to Understand that the Parametrized Constructor of the Super Class can be called from Sub Class Using super().

A subclass can call a constructor defined by its superclass by use of the following form of super: super(arg-list); Here, arg-list specifies any arguments needed by the constructor in the superclass.

Here is the source code of the Java Program to Understand that the Parametrized Constructor of the Super Class can be called from Sub Class Using super(). The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class sup1
3. {
4.     int z;
5.     sup1(int x, int y)
6.     {
7.         z = x + y;
8.         System.out.println("Result:" + z);
9.     }
10.}
11. public class Parametrized_Constructor_Demo extends sup1
12. {
13.     Parametrized_Constructor_Demo(int x, int y)
14.     {
15.         super(x, y);
16.     }
17.     public static void main(String[] args)
18.     {
19.         Scanner s = new Scanner(System.in);
20.         System.out.print("Enter value of a:");
21.         int a = s.nextInt();
22.         System.out.print("Enter value of b:");
23.         int b = s.nextInt();
24.         Parametrized_Constructor_Demo obj = new Parametrized_Constructor_Demo(a, b);
25.     }
26. }
```

Output:

```
$ javac Parametrized_Constructor_Demo.java
$ java Parametrized_Constructor_Demo
```

```
Enter value of a:32
Enter value of b:23
Result:55
```

90. Java Program to Illustrate Use of All Features of Abstract Class

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate Use of All Features of Abstract Class.

Sometimes you will want to create a superclass that only defines a generalized form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details. Such a class determines the nature of the methods that the subclasses must implement.

Here is the source code of the Java Program to Illustrate Use of All Features of Abstract Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. abstract class Operations
2. {
3.     float a = 12, b = 6, c;
4.     abstract void add();
5.     void subtract()
6.     {
7.         c = a - b;
8.         System.out.println("Result:"+c);
9.     }
10.    abstract void multiply();
11.    void divide()
12.    {
13.        c = a / b;
14.        System.out.println("Result:"+c);
15.    }
16.
17.public class Abstract_Demo extends Operations
18.{
19.    @Override
20.    void add()
21.    {
22.        c = a + b;
23.        System.out.println("Result:"+c);
24.    }
25.    @Override
26.    void multiply()
27.    {
28.        c = a * b;
29.        System.out.println("Result:"+c);
30.    }
31.    public static void main(String[] args)
32.    {
33.        Abstract_Demo obj = new Abstract_Demo();
34.        obj.add();
35.        obj.subtract();
36.        obj.multiply();
37.        obj.divide();
38.    }
39.}
```

Output:

```
$ javac Abstract_Demo.java
$ java Abstract_Demo
```

```
Result:18.0
Result:6.0
Result:72.0
```

Result:2.0

92. Java Program to Illustrate Use of Chaining Constructor

[« Prev](#)

[Next »](#)

This is a Java Program to illustrate the use of chaining of constructor.

Calling of one constructor from another constructor with respect to current object is constructor chaining. It can be achieved in any order.

Here is the source code of the Java Program to illustrate the use of chaining of constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Constructor_Chaining
2. {
3.     public Constructor_Chaining
4.     {
5.         System.out.println("In default constructor");
6.     }
7.     public Constructor_Chaining(int i)
8.     {
9.         this();
10.        System.out.println("In single parameter constructor");
11.    }
12.    public Constructor_Chaining(int i,int j)
13.    {
14.        this(j);
15.        System.out.println("In double parameter constructor");
16.    }
17.    public static void main(String a[])
18.    {
19.        Constructor_Chaining obj = new Constructor_Chaining(11,12);
20.    }
21.}
```

Output:

```
$ javac Constructor_Chaining.java
$ java Constructor_Chaining
```

```
In default constructor
In single parameter constructor
In double parameter constructor
```

93. Java Program to Create the Object for Class and to Assign Value in the Object Using Constructor

[« Prev](#)

[Next »](#)

This is a Java Program to Create the Object for Class and to Assign Value in the Object Using Constructor. Constructor is used to initialise the object. A constructor initializes an object immediately upon creation.

Here is the source code of the Java Program to Create the Object for Class and to Assign Value in the Object Using Constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Demo
2. {
3.     int a,b,c;
4.     Demo()
5.     {
6.         a = 5;
7.         b = 6;
8.     }
9.     void change()
10.    {
11.        a = 15;
12.        b = 10;
13.        c = a + b;
14.    }
15.    public static void main(String[] args)
16.    {
17.        Demo obj1 = new Demo();
18.        System.out.println("a:"+obj1.a);
19.        System.out.println("b:"+obj1.b);
20.        obj1.change();
21.        System.out.println("New a:"+obj1.a);
22.        System.out.println("New b:"+obj1.b);
23.        System.out.println("c:"+obj1.c);
24.    }
25.}
```

Output:

```
$ javac Demo.java
$ java Demo
```

```
a:5
b:6
New a:15
New b:10
c:25
```

94. Java Program to Allocate and Initialize Super Class Members Using Constructor

[« Prev](#)

[Next »](#)

This is a Java Program to Allocate and Initialize Super Class Members Using Constructor. In a class hierarchy, constructors are called in order of derivation, from superclass to subclass.

Here is the source code of the Java Program to Allocate and Initialize Super Class Members Using Constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Parent
2. {
3.     Parent(int a, int b)
4.     {
5.         System.out.println(" the super class constructor");
6.         int z = a + b;
7.         System.out.println("the super class method ");
8.         System.out.println("the sum is "+z);
9.     }
10. }
11.public class Child extends Parent
12.
13. Child(int x)
14. {
15.     super(12, 20);
16.     System.out.println("the sub class constructor ");
17.     System.out.println(x);
18. }
19. public static void main(String ... a)
20. {
21.     Child obj = new Child(10);
22. }
23.}
```

Output:

```
$ javac Child.java
$ java Child
```

```
the super class constructor
the super class method
the sum is 32
the sub class constructor
10
```

95. Java Program to Create Strings and How to Use Some Important Methods of String Class

[« Prev](#)

[Next »](#)

This is a Java Program to Create Strings and How to Use Some Important Methods of String Class.

The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. There are some examples to show use of some methods of String.

Here is the source code of the Java Program to Create Strings and How to Use Some Important Methods of String Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Important_Methods
2. {
3.     public static void main(String... a)
4.     {
5.         String str1 = "Hello";
6.         String str2 = "from";
7.         String str3 = "JAVA";
8.         int len = str1.length();
9.         System.out.println("The length of the String str1 using length() is "+len);
10.        String str4 = str1.concat(str2).concat(str3);
11.        System.out.println("The string after concatenation using concat() is " +str4);
12.        char c = str4.charAt(5);
13.        System.out.println("The character atindex 5 using charAt() is " +c);
14.        String str5 = "AbCdEfGhIjKlMnOpQrStUvWxYz";
15.        String str6[] = str5.split("M");
16.        String part1 = str6[0];
17.        String part2 = str6[1];
18.        System.out.println("The string after splitting using split() will be = "+part1+" "+part2);
19.    }
20.}
```

Output:

```
$ javac Important_Methods.java
$ java Important_Methods
```

```
The length of the String str1 using length() is 5
The string after concatenation using concat() is HellofromJAVA
The character atindex 5 using charAt() is f
The string after splitting using split() will be = AbCdEfGhIjKl and nOpQrStUvWxYz
```

96. Java Program to Check Whether Which One is Executed First, Static block or the Static Method

[« Prev](#)

[Next »](#)

This is a Java Program to Check Whether Which One is Executed First , Static block or the Static Method. Static block is used to initialise the static data members dynamically. If you want to perform a task only once in life cycle of class then put that code in static block.

Here is the source code of the Java Program to Check Whether Which One is Executed First , Static block or the Static Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Demo
2. {
3.     static
4.     {
5.         System.out.println("First static block");
6.     }
7.
8.     public Demo()
9.     {
10.        System.out.println("Constructor");
11.    }
12.
13.    public static String staticString = "Static Variable";
14.
15.    static
16.    {
17.        System.out.println("Second static block and "+ staticString);
18.    }
19.    static
20.    {
21.        staticMethod();
22.        System.out.println("Third static block");
23.    }
24.
25.    public static void staticMethod()
26.    {
27.        System.out.println("Static method");
28.    }
29.
30.    public static void staticMethod2()
31.    {
32.        System.out.println("Static method2");
33.    }
34.    public static void main(String[] args)
35.    {
36.        Demo obj = new Demo();
37.        obj.staticMethod2();
38.    }
39.}
```

Output:

```
$ javac Demo.java
$ java Demo
```

First static block

Second static block and Static Variable

Static method

Third static block

Constructor

Static method2

97. Java Program to Interchange Two Employee Objects by Passing them to Swap() Method

[« Prev](#)

[Next »](#)

This is a Java Program to Interchange Two Employee Objects by Passing them to Swap() Method.

Here we make a swap method to interchange the information of two employees using Employee Objects by passing them to swap() method.

Here is the source code of the Java Program to Interchange Two Employee Objects by Passing them to Swap() Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Swap_Demo
2. {
3.     static String emp1, emp2;
4.     Swap_Demo()
5.     {
6.         emp1 = "this is A";
7.         emp2 = "this is B";
8.     }
9.     void swap(String x, String y)
10.    {
11.        String res;
12.        res = x;
13.        x = y;
14.        y = res;
15.        System.out.println("Employee1: "+x);
16.        System.out.println("Employee2: "+y);
17.    }
18.    public static void main(String[] args)
19.    {
20.        Swap_Demo obj = new Swap_Demo();
21.        System.out.println("Before swapping:");
22.        System.out.println("Employee1: "+emp1);
23.        System.out.println("Employee2: "+emp2);
24.        System.out.println("After swapping:");
25.        obj.swap(emp1, emp2);
26.    }
27.}
```

Output:

```
$ javac Swap_Demo.java
$ java Swap_Demo
```

```
Before swapping:
Employee1:this is A
Employee2:this is B
After swapping:
Employee1: this is B
Employee2: this is A
```

98. Java Program Which has 2 Classes Which Initializes a String in Its Constructor

[« Prev](#)

[Next »](#)

This is a Java Program Which has two Classes Which Initializes a String in Its Constructor.

Constructor is used to initialise an object. Here we made two classes as Super and Sub class using Inheritance. We initialise two different strings in the constructors of these two classes and with the help of object of Sub class we print these strings.

Here is the source code of the Java Program Which has two Classes Which Initializes a String in Its Constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class String_Initialise
2. {
3.     String a, b;
4.     public String_Initialise()
5.     {
6.         System.out.println("Base Class Constructor");
7.         a = "String from Base Class";
8.     }
9. }
10. public class String_Initialise1 extends String_Initialise
11. {
12.     public String_Initialise1()
13.     {
14.         System.out.println("Derived Class Constructor");
15.         b = "String from Derived Class";
16.     }
17.     public static void main(String... arg)
18.     {
19.         String_Initialise1 obj = new String_Initialise1();
20.         System.out.println("the strings initialised in the constructors of Base and Derived classes are :");
21.         System.out.println(obj.a + " and " + obj.b);
22.     }
23. }
```

Output:

```
$ javac String_Initialise1.java
$ java String_Initialise1
```

```
Base Class Constructor
Derived Class Constructor
the strings initialised in the constructors of Base and Derived classes are :
String from Base Class and String from Derived Class
```

99. Java Program to Make Shape as an Interface and Implement it using Circle and Rectangle Class

[« Prev](#)

[Next »](#)

This is a Java Program to Make Shape as an Interface and Implement it using Circle and Rectangle Class.

Interfaces are syntactically similar to classes, but they lack instance variables, and their methods are declared without any body. Here we make interface as Shape with two methods as input() and area() which are implemented by further two classes as circle and rectangle who implements the interface Shape.

Here is the source code of the Java Program to Make Shape as an Interface and Implement it using Circle and Rectangle Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. interface Shape
2. {
3.     void input();
4.     void area();
5. }
6. class Circle implements Shape
7. {
8.     int r = 0;
9.     double pi = 3.14, ar = 0;
10.    @Override
11.    public void input()
12.    {
13.        r = 5;
14.    }
15.    @Override
16.    public void area()
17.    {
18.        ar = pi * r * r;
19.        System.out.println("Area of circle:"+ar);
20.    }
21.}
22.class Rectangle extends Circle
23.{
24.    int l = 0, b = 0;
25.    double ar;
26.    public void input()
27.    {
28.        super.input();
29.        l = 6;
30.        b = 4;
31.    }
32.    public void area()
33.    {
34.        super.area();
35.        ar = l * b;
36.        System.out.println("Area of rectangle:"+ar);
37.    }
38.}
39.public class Demo
40.{
41.    public static void main(String[] args)
42.    {
43.        Rectangle obj = new Rectangle();
44.        obj.input();
45.        obj.area();
46.    }
47.}
```

Output:

```
$ javac Demo.java  
$ java Demo
```

```
Area of circle:78.5  
Area of rectangle:24.0
```

100. Java Program to Split into Pieces Wherever a Space is Found

[« Prev](#)

[Next »](#)

This is a Java Program to Split into Pieces Wherever a Space is Found.

This is one of the important methods of String. `split(String regex)` :- Splits this string around matches of the given regular expression.

Here is the source code of the Java Program to Split into Pieces Wherever a Space is Found. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Split_Demo
2. {
3.     public static void main(String... a)
4.     {
5.         String s1 = "HelloJava from JavaGuy";
6.         String s2[] = s1.split(" ");
7.         String part1 = s2[0];
8.         String part2 = s2[1];
9.         String part3 = s2[2];
10.        System.out.println("The string after splitting is "+part1+ " and "+part2);
11.    }
12.}
```

Output:

```
$ javac Split_Demo.java
$ java Split_Demo
```

The string after splitting is Hello and Java

101. Java Program to Use This Keyword in Inheritance Class

[« Prev](#)

[Next »](#)

This is a Java Program to Use Super Keyword in Inheritance Class.

Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super. We made two same data member in base and child class and we call the base class data member from child class by using Super Keyword.

Here is the source code of the Java Program to Use Super Keyword in Inheritance Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Base
2. {
3.     int x = 19;
4. }
5.
6. class Child extends Base
7. {
8.     int x = 20;
9.     void shows()
10.    {
11.        System.out.println("The base class data member (x) by Super Keyword :" + super.x);
12.        System.out.println("The child class data member :" + x);
13.
14.    }
15. public static void main(String... a)
16. {
17.     Child obj = new Child();
18.     obj.shows();
19. }
20. }
```

Output:

```
$ javac Child.java
$ java Child
```

```
The base class data member (x)by Super Keyword :19
The child class data member :20
```

102. Java Program to Use Super Keyword in Inheritance Class

[« Prev](#)

[Next »](#)

This is a Java Program to Use This Keyword in Inheritance Class.

In multi level inheritance we can only call the data member of parent class from child class via Super keyword but we cannot call the data member of grand parent class from child class. We can achieve this via This keyword. Here we are doing the same to print the value of X of grand parent class from child class.

Here is the source code of the Java Program to Use This Keyword in Inheritance Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Base1
2. {
3.     int x = 19;
4. }
5. class Base2 extends Base1
6. {
7.     int x = 20;
8. }
9. class Child extends Base2
10. {
11.     int x = 21;
12.     void show()
13.     {
14.         System.out.println("Child Class-"+x);
15.         System.out.println("Base1 class-"+super.x);
16.         System.out.println("Accessing value of data member via This Keyword");
17.         System.out.println("Base2 class-"+(((Base1)this).x));
18.     }
19.     public static void main(String... a)
20.     {
21.         Child obj = new Child();
22.         obj.show();
23.     }
24. }
```

Output:

```
$ javac Child.java
$ java Child
```

```
Child Class-21
Base1 class-20
Accessing value of data member via This Keyword
Base2 class-19
```

103. Java Program to Show Method Overriding in a Class Using Inheritance Class

[« Prev](#)

[Next »](#)

This is a Java Program to Show Method Overriding in a Class Using Inheritance Class.

In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass. Here we made showme() method in both classes and achieve the concept of Method Overriding.

Here is the source code of the Java Program to Show Method Overriding in a Class Using Inheritance Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Base
2. {
3.     void showme()
4.     {
5.         System.out.println(" Base class method");
6.     }
7. }
8. class Child extends Base
9. {
10.    void showme()
11.    {
12.        System.out.println("Child class method");
13.    }
14. public static void main(String... a)
15. {
16.     Child obj = new Child();
17.     obj.showme();
18. }
19. }
```

Output:

```
$ javac Child.java
$ java Child
```

```
Child class method
```

104. Java Program to Access Super Class in a Method Overriding

[« Prev](#)

[Next »](#)

This is a Java Program to Access Super Class in a Method Overriding.

Here we made get() method in both base and child class. Child class override the method of base class and to run both methods we use the Super keyword.

Here is the source code of the Java Program to Access Super Class in a Method Overriding. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Base
2. {
3.     void get()
4.     {
5.         System.out.println(" Base class method via Super keyword");
6.     }
7. }
8. class Child extends Base
9. {
10.    void get()
11.    {
12.        super.get();
13.        System.out.println("Child class method");
14.    }
15. public static void main(String... a)
16. {
17.     Child obj1 = new Child();
18.     obj1.get();
19. }
20. }
```

Output:

```
$ javac Child.java
$ java Child
```

```
Base class method via Super keyword
Child class method
```

105. Java Program to Find Season Using Switch Statement

[« Prev](#)

[Next »](#)

This is a Java Program to Find Season Using Switch Statement. The switch statement is Java's multi way branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression.

We take the month as an input. The value of the input is compared with each of the literal values in the case statements. If a match is found, the code sequence following that case statement is executed. If none of the constants matches the value of the expression, then the default statement is executed.

Here is the source code of the Java Program to Find Season Using Switch Statement. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Switch_Demo
3. {
4.     public static void main(String args[])
5.     {
6.         String season;
7.         System.out.println("Enter any month(1 to 12)");
8.         Scanner s = new Scanner(System.in);
9.         int entry = s.nextInt();
10.        switch (entry)
11.        {
12.            case 12:
13.            case 1:
14.            case 2:
15.                season = "Winter";
16.                break;
17.            case 3:
18.            case 4:
19.            case 5:
20.                season = "Spring";
21.                break;
22.            case 6:
23.            case 7:
24.            case 8:
25.                season = "Summer";
26.                break;
27.            case 9:
28.            case 10:
29.            case 11:
30.                season = "Autumn";
31.                break;
32.            default:
33.                season = "Bogus Month";
34.        }
35.        System.out.println("The entered month is in the " + season + ".");
36.    }
37.}
```

Output:

```
$ javac Switch_Demo.java
$ java Switch_Demo
```

```
Enter any month(1 to 12)
3
The entered month is in the Spring.
```

106. Java Program to Add Two Complex Numbers

« [Prev](#)

[Next](#) »

This is the Java Program to Add Two Complex Numbers.

Problem Description

Given two complex numbers, write a java program to add the two numbers.

Problem Solution

Add the real and imaginary parts of the numbers seperately and store them in a new variable.
advertisement

Program/Source Code

Here is the source code of the Java Program to Add Two Complex Numbers. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1. 
2. //Java Program to Add Two Complex Numbers
3.
4. import java.io.BufferedReader;
5. import java.io.InputStreamReader;
6.
7. public class ComplexNumbers {
8.     // Main function to read two complex numbers and add them
9.     public static void main(String[] args) {
10.         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11.         double i1,j1,i2,j2;
12.         System.out.println("Enter the real part and
13.                         imaginary part of the first complex number");
14.         try{
15.             i1=Double.parseDouble(br.readLine());
16.             j1=Double.parseDouble(br.readLine());
17.         }catch (Exception e){
18.             System.out.println("An error occurred");
19.             return;
20.         }
21.         System.out.println("Enter the real part and
22.                         imaginary part of the second complex number");
23.         try{
24.             i2=Double.parseDouble(br.readLine());
25.             j2=Double.parseDouble(br.readLine());
26.         }catch (Exception e){
27.             System.out.println("An error occurred");
28.             return;
29.         }
30.         System.out.println("The first complex number is "
31.                         + i1 + " + i(" + j1 + ")");
32.         System.out.println("The second complex number is "
33.                         + i2 + " + i(" + j2 + ")");
34.         System.out.println("The sum of the two complex numbers is "
35.                         + (i1 + i2) + " + i(" + (j1 + j2) + ")");
36.     }
37. }
```

Program Explanation

1. In function main(), the real parts of the complex number are stored in i1 and i2 and the imaginary parts are stored in j1 and j2.
2. Then, in the third println statement real parts are added separately and imaginary parts are added separately.

Time Complexity: O(1).

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the real part and imaginary part of the first complex number

4

6

Enter the real part and imaginary part of the second complex number

-5

2

The first complex number is $4.0 + i(6.0)$

The second complex number is $-5.0 + i(2.0)$

The sum of the two complex numbers is $-1.0 + i(8.0)$

107. Java Program to Display Pascal Triangle

« [Prev](#)

[Next](#) »

This is the Java program to print the Pascal's Triangle of a given size.

Problem Description

Pascal's triangle is an matrix of the binomial coefficients. The number of entries in every row increases by 1 starting from the first row having 1 entry and the second row having 2 entries and so on.

Example:

```
1  
1 1  
1 2 1  
1 3 3 1
```

Problem Solution

For every row of the matrix calculate the binomial coefficients $C(\text{row},0)$ to $C(\text{row}, \text{row})$ and display them. The binomial coefficient can be calculated using the formula:

advertisement

$$C(n,r) = n! / ((n-r)! * r!)$$

Program/Source Code

Here is the source code of the Java Program to print the Pascal's Triangle of a given size. The program is successfully compiled and tested using the IDE IntelliJ Idea in Windows 7. The program output is also shown below.

```
1.  
2. //Java program to print Pascal's Triangle of a given size.  
3.  
4. import java.io.BufferedReader;  
5. import java.io.InputStreamReader;  
6.  
7. public class PascalTriangle {  
8.     // Function to calculate factorial of a number  
9.     static int factorial(int n)  
10.    {  
11.        int fact = 1;  
12.        int i;  
13.        for(i=1; i<n; i++)  
14.        {  
15.            fact*=i;  
16.        }  
17.        return i;  
18.    }  
19.    // Function to display the pascal triangle  
20.    static void display(int n)  
21.    {  
22.        int i;  
23.        int coefficient;  
24.        int line;  
25.        for(line=1;line<=n;line++)  
26.        {  
27.            for(i=0;i<=line;i++)  
28.            {  
29.                System.out.print((factorial(line)/factorial(line-i) * factorial(i)) + " ");  
30.            }  
31.            System.out.println();
```

```

32. }
33. }
34. //main Function to read user input
35. public static void main(String[] args){
36.     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
37.     int n;
38.     System.out.println("Enter the size");
39.     try {
40.         n = Integer.parseInt(br.readLine());
41.     }
42.     catch(Exception e){
43.         System.out.println("Invalid Input");
44.         return;
45.     }
46.     System.out.println("The Pascal's Triangle is");
47.     display(n);
48. }
49.

```

Program Explanation

1. In function display(), the loop for(line=1;line<=n;line++) is used to print n lines of the triangle.
2. The nested loop for(i=1;i<=line;i++) is used to display various coefficients.
3. The various coefficients are calculated using the factorial function, as mentioned in the problem solution.

advertisement

Time Complexity: $O(n^2)$ where n is the number of lines.

Runtime Test Cases

Case 1 (Simple Test Case):

Enter the size

5

The Pascal's Triangle is

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

Case 2 (Simple Test Case - another example):

Enter the size

12

The Pascal's Triangle is

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1

```

108. Java Program to Display Floyd's Triangle

« [Prev](#)

[Next](#) »

This is a Java Program to Display Floyd's Triangle.

In Floyd triangle there are n integers in the nth row and a total of $(n(n+1))/2$ integers in n rows. We take input as number of rows user want to print.

advertisement

Here is the source code of the Java Program to Display Floyd's Triangle. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. import java.util.Scanner;
2.
3. class Floyd
4. {
5.     public static void main(String args[])
6.     {
7.         int n, num = 1, c, d;
8.         Scanner in = new Scanner(System.in);
9.         System.out.println("Enter the number of rows of floyd's triangle you want");
10.        n = in.nextInt();
11.        System.out.println("Floyd's triangle :-");
12.        for (c = 1; c <= n; c++)
13.        {
14.            for (d = 1; d <= c; d++)
15.            {
16.                System.out.print(num + " ");
17.                num++;
18.            }
19.            System.out.println();
20.        }
21.    }
22. }
```

Output:

advertisement

```
$ javac Floyd.java
$ java Floyd
```

Enter the number of rows of floyd's triangle you want

```
5
Floyd's triangle :-  
1  
2 3  
4 5 6  
7 8 9 10  
11 12 13 14 15
```

109. Java Program to Generate Harmonic Series

[« Prev](#)

[Next »](#)

This is a Java Program to Generate Harmonic Series.

We take a number as an input and using loops we generate the Harmonic series and get the desired output.

Here is the source code of the Java Program to Generate Harmonic Series. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Harmonic
3. {
4. public static void main(String... a)
5. {
6.     System.out.print("Enter any number : ");
7.     Scanner s = new Scanner(System.in);
8.     int num = s.nextInt();
9.     System.out.print("The Harmonic Series is : ");
10.    double result = 0.0;
11.    while(num > 0)
12.    {
13.        result = result + (double) 1 / num;
14.        num--;
15.        System.out.print(result + " ");
16.    }
17.    System.out.println("");
18.    System.out.println("Output of Harmonic Series is "+result);
19. }
20. }
```

Output:

```
$ javac Harmonic.java
$ java Harmonic
```

```
Enter any number : 5
The Harmonic Series is : 0.2 0.45 0.7833333333333333 1.283333333333332 2.283333333333333
Output of Harmonic Series is 2.283333333333333
```

110. Java Program to Convert Hex to Decimal

« [Prev](#)

[Next](#) »

This is a Java Program to Convert Hex to Decimal.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as HexaDecimal from user. We define a method as convert() to convert the user's input to Decimal.

Here is the source code of the Java Program to Convert Hex to Decimal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Hexa.Decimal
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("HexaDecimal to Decimal");
9.         scan = new Scanner(System.in);
10.        System.out.println("Enter the number :");
11.        num = Integer.parseInt(scan.nextLine(), 16);
12.    }
13.    void convert()
14.    {
15.        String decimal = Integer.toString(num);
16.        System.out.println("Decimal Value is : " + decimal);
17.    }
18.
19.class Hexa.Decimal_Main
20.
21.    public static void main(String args[])
22.    {
23.        Hexa.Decimal obj = new Hexa.Decimal();
24.        obj.getVal();
25.        obj.convert();
26.    }
27.}
```

Output:

```
$ javac Hexa.Decimal_Main.java
$ java Hexa.Decimal_Main
```

```
HexaDecimal to Decimal
Enter the number :
12
Decimal Value is : 18
```

111. Java Program to Convert Decimal to Binary, Octal and Hexadecimal Number

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Decimal to Binary, Octal and Hexadecimal Number.

Here we take input as Decimal number and convert this entered number to HexaDecimal, Octal and Binary using methods toHexString(), toOctalString() and toBinaryString() respectively.

Here is the source code of the Java Program to Convert Decimal to Binary, Octal and Hexadecimal Number. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. class Convert
2. {
3.     Scanner scan;
4.     int num;
5.     void getVal()
6.     {
7.         System.out.println("Decimal to HexaDecimal,Octal and Binary");
8.         scan = new Scanner(System.in);
9.         System.out.println("\nEnter the number :");
10.        num = Integer.parseInt(scan.nextLine());
11.    }
12.    void convert()
13.    {
14.        String hexa = Integer.toHexString(num);
15.        System.out.println("HexaDecimal Value is : " + hexa);
16.        String octal = Integer.toOctalString(num);
17.        System.out.println("Octal Value is : " + octal);
18.        String binary = Integer.toBinaryString(num);
19.        System.out.println("Binary Value is : " + binary);
20.    }
21.}
22.class Decimal_Conversion
23{
24.    public static void main(String args[])
25.    {
26.        Convert obj = new Convert();
27.        obj.getVal();
28.        obj.convert();
29.    }
30.}
```

Output:

```
$ javac Decimal_Conversion.java
$ java Decimal_Conversion
```

```
Decimal to HexaDecimal,Octal and Binary
```

```
Enter the number :
```

```
121
```

```
HexaDecimal Value is : 79
```

```
Octal Value is : 171
```

```
Binary Value is : 1111001
```

112. Java Program to Convert Binary to Octal

« [Prev](#)

[Next](#) »

This is a Java Program to Convert Binary to Octal.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as Binary from user. We define a method as convert() to convert the user's input to Octal.

Here is the source code of the Java Program to Convert Binary to Octal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Binary_Octal
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("Binary to Octal");
9.         scan = new Scanner(System.in);
10.        System.out.println("\nEnter the number :");
11.        num = Integer.parseInt(scan.nextLine(), 2);
12.    }
13.    void convert()
14.    {
15.        String octal = Integer.toOctalString(num);
16.        System.out.println("Octal Value is : " + octal);
17.    }
18. }
19.class Main_Class
20.
21. public static void main(String... d)
22. {
23.     Binary_Octal obj = new Binary_Octal();
24.     obj.getVal();
25.     obj.convert();
26. }
27. }
```

Output:

```
$ javac Main_Class.java
$ java Main_Class
```

Binary to Octal

Enter the number :

1010

Octal Value is : 12

113. Java Program to Convert Binary to Hexadecimal

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Binary to Hexadecimal.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as Binary from user. We define a method as convert() to convert the user's input to HexaDecimal.

Here is the source code of the Java Program to Convert Binary to Hexadecimal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Binary_Hexa
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("Binary to HexaDecimal");
9.         scan = new Scanner(System.in);
10.        System.out.println("\nEnter the number :");
11.        num = Integer.parseInt(scan.nextLine(), 2);
12.    }
13.    void convert()
14.    {
15.        String hexa = Integer.toHexString(num);
16.        System.out.println("HexaDecimal Value is : " + hexa);
17.    }
18. }
19.class Main_Class
20.
21. public static void main(String... q)
22. {
23.     Binary_Hexa obj = new Binary_Hexa();
24.     obj.getVal();
25.     obj.convert();
26. }
27.}
```

Output:

```
$ javac Main_Class.java
$ java Main_Class
```

```
Binary to HexaDecimal
```

```
Enter the number :
```

```
1010
```

```
HexaDecimal Value is : a
```

114. Java Program to Convert Octal to Binary

« [Prev](#)

[Next](#) »

This is a Java Program to Convert Octal to Binary.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as Octal from user. We define a method as convert() to convert the user's input to Binary.

Here is the source code of the Java Program to Convert Octal to Binary. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Octal_Binary
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("Octal to Binary");
9.         scan = new Scanner(System.in);
10.
11.        System.out.println("\nEnter the number :");
12.        num = Integer.parseInt(scan.nextLine(), 8);
13.    }
14.
15. void convert()
16. {
17.     String binary = Integer.toBinaryString(num);
18.     System.out.println("Binary Value is : " + binary);
19. }
20.
21.class MainClass
22.
23. public static void main(String args[])
24. {
25.     Octal_Binary obj = new Octal_Binary();
26.     obj.getVal();
27.     obj.convert();
28. }
29.
```

Output:

```
$ javac MainClass.java
$ java MainClass
```

```
Octal to Binary
```

```
Enter the number :
```

```
10
```

```
Binary Value is : 1000
```

115. Java Program to Convert Hexadecimal to Binary

[« Prev](#)

[Next »](#)

This is a Java Program to Convert Hexadecimal to Binary.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as HexaDecimal from user. We define a method as convert() to convert the user's input to Binary.

Here is the source code of the Java Program to Convert Hexadecimal to Binary. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Hexa_Binary
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("HexaDecimal to Binary");
9.         scan = new Scanner(System.in);
10.        System.out.println("\nEnter the number :");
11.        num = Integer.parseInt(scan.nextLine(), 16);
12.    }
13.    void convert()
14.    {
15.        String binary = Integer.toBinaryString(num);
16.        System.out.println("Binary Value is : " + binary);
17.    }
18. }
19.class MainClass
20{
21.    public static void main(String args[])
22.    {
23.        Hexa_Binary obj = new Hexa_Binary();
24.        obj.getVal();
25.        obj.convert();
26.    }
27.}
```

Output:

```
$ javac MainClass.java
$ java MainClass
```

```
HexaDecimal to Binary
```

```
Enter the number :
```

```
20
```

```
Binary Value is : 100000
```

116. Java Program to Convert Octal to Decimal

« [Prev](#)

[Next](#) »

This is a Java Program to Convert Octal to Decimal.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as Octal from user. We define a method as convert() to convert the user's input to Decimal.

Here is the source code of the Java Program to Convert Octal to Decimal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Octal.Decimal
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("Octal to Decimal");
9.         scan = new Scanner(System.in);
10.        System.out.println("\nEnter the number :");
11.        num = Integer.parseInt(scan.nextLine(), 8);
12.    }
13.    void convert()
14.    {
15.        String decimal = Integer.toString(num);
16.        System.out.println("Decimal Value is : " + decimal);
17.    }
18.
19.class MainClass
20.
21. public static void main(String args[])
22. {
23.     Octal.Decimal obj = new Octal.Decimal();
24.     obj.getVal();
25.     obj.convert();
26. }
27.
```

Output:

```
$ javac MainClass.java
$ java MainClass
```

Octal to Decimal

```
Enter the number :
10
```

```
Decimal Value is : 8
```

117. Java Program to Convert the Given Binary Number into its Equivalent Decimal

[« Prev](#)

[Next »](#)

This is a Java Program to Convert the Given Binary Number into its Equivalent Decimal.

We make a class with two methods one for input and other for conversion and access this by object of this class. We first take the input as Binary from user. We define a method as convert() to convert the user's input to Decimal.

Here is the source code of the Java Program to Convert the Given Binary Number into its Equivalent Decimal. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. class Binary_Decimal
3. {
4.     Scanner scan;
5.     int num;
6.     void getVal()
7.     {
8.         System.out.println("Binary to Decimal");
9.         scan = new Scanner(System.in);
10.    System.out.println("\nEnter the number :");
11.    num = Integer.parseInt(scan.nextLine(), 2);
12. }
13. void convert()
14. {
15.     String decimal = Integer.toString(num);
16.     System.out.println("Decimal Value is : " + decimal);
17. }
18.
19.class MainClass
20.
21. public static void main(String args[])
22. {
23.     Binary_Decimal obj = new Binary_Decimal();
24.     obj.getVal();
25.     obj.convert();
26. }
27. }
```

Output:

```
$ javac MainClass.java
$ java MainClass
```

```
Binary to Decimal
```

```
Enter the number :
```

```
1010
```

```
Decimal Value is : 10
```

118. Java Program to Segregate 0s on Left Side & 1s on Right Side of the Array

[« Prev](#)

[Next »](#)

This is a Java Program to Segregate 0s on Left Side & 1s on Right Side of the Array.

Here is a function to segregate all 0s on left and all 1s on right as segregate0and1. We define an array and pass this array along with its length to the function and segregate 0s and 1s using this function.

Here is the source code of the Java Program to Segregate 0s on Left Side & 1s on Right Side of the Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class ArraySegregate
2. {
3.     public static void main(String... a)
4.     {
5.         int array[] = { 0, 1, 0, 1, 1, 0 };
6.         segregate0and1(array, 6);
7.         for(int i = 0 ; i < 6; i++)
8.         {
9.             System.out.print(array[i]+"\t");
10.        }
11.    }
12.    static void segregate0and1(int array[], int size)
13.    {
14.        int left = 0, right = size-1;
15.        while (left < right)
16.        {
17.            /* Increment left index while we see 0 at left */
18.            while (array[left] == 0 && left < right)
19.                left++;
20.            /* Decrement right index while we see 1 at right */
21.            while (array[right] == 1 && left < right)
22.                right--;
23.            /* If left is smaller than right then there is a 1 at left and a 0 at right. Exchange it */
24.            if (left < right)
25.            {
26.                array[left] = 0;
27.                array[right] = 1;
28.                left++;
29.                right--;
30.            }
31.        }
32.    }
33.}
```

Output:

```
$ javac ArraySegregate.java
$ java ArraySegregate
```

```
0    0    0    1    1    1
```

119. Java Program to Identify Missing Numbers in a Given Array

[« Prev](#)

[Next »](#)

This is a Java Program to Identify Missing Numbers in a Given Array.

We made a method as getMissingNo in which we pass the array and length of the array as arguments and using this method we calculate the missing number and hence we get the desired output.

Here is the source code of the Java Program to Identify Missing Numbers in a Given Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Missing
2. {
3.     static int getMissingNo (int a[], int n)
4.     {
5.         int i, total;
6.         total = (n + 1) * (n + 2) / 2;
7.         for ( i = 0; i < n; i++)
8.             total -= a[i];
9.         return total;
10.    }
11.    public static void main(String... s)
12.    {
13.        int a[ ] = {1, 2, 4, 5, 6};
14.        int miss = getMissingNo(a, 5);
15.        System.out.println("The number missing is :" + miss);
16.    }
17.}
```

Output:

```
$ javac Missing.java
$ java Missing
```

The number missing is :3

120. Java Program to find GCD and LCM of Two Numbers Using Euclid's Algorithm

[« Prev](#)

[Next »](#)

This is a Java Program to find GCD and LCM of Two Numbers Using Euclid's Algorithm. Euclid's Algorithm is an algorithm, a step-by-step procedure for performing a calculation according to well-defined rules, and is one of the oldest numerical algorithms in common use.

We divide the two numbers entered by the user and remainder become divisor while previous divisor become dividend now. This process repeat until the remainder become zero and we get the GCD as divisor which gives remainder as zero. LCM is a little trickier, but probably the best approach is reduction by the GCD, which can be similarly iterated.

Here is the source code of the Java Program to find GCD and LCM of Two Numbers Using Euclid's Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.Scanner;
2. public class Euclid
3. {
4.     void gcd(long a, long b)
5.     {
6.         while (b > 0)
7.         {
8.             long temp = b;
9.             b = a % b; // % is remainder
10.            a = temp;
11.        }
12.        System.out.println("GCD is "+a);
13.    }
14.    void lcm(long a, long b)
15.    {
16.        long x = a;
17.        long y = b;
18.        while (b > 0)
19.        {
20.            long temp = b;
21.            b = a % b; // % is remainder
22.            a = temp;
23.        }
24.        long gcd = a;
25.        long lcm = (x * (y / gcd));
26.        System.out.println("LCM is "+lcm);
27.    }
28.    public static void main(String... a)
29.    {
30.        Euclid abc = new Euclid();
31.        System.out.println("Enter any two numbers to calculate GCD");
32.        Scanner s = new Scanner(System.in);
33.        long x = s.nextLong();
34.        long y = s.nextLong();
35.        abc.gcd(x, y);
36.        System.out.println("Enter any two numbers to calculate LCM");
37.        long l = s.nextLong();
38.        long m = s.nextLong();
39.        abc.lcm(l, m);
40.    }
41.}
```

Output:

```
$ javac Euclid.java  
$ java Euclid
```

Enter any two numbers to calculate GCD

6 50

GCD is 2

Enter any two numbers to calculate LCM

11 17

LCM is 187

121. Java Program to Illustrate the Use of HashCode() Method

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrate the Use of HashCode() Method. The hashCode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure.

We have to implement hashCode() method of a class in such way that if two objects are equals, ie compared by equals() method of that class, then those two objects must return same hash code. The below example shows how to override equals and hashCode methods. The class Price overrides equals and hashCode. If you notice the hashCode implementation, it always generates unique hashCode for each object based on their state, ie if the object state is same, then you will get same hashCode. A HashMap is used in the example to store Price objects as keys. It shows though we generate different objects, but if state is same, still we can use this as key.

Here is the source code of the Java Program to Illustrate the Use of HashCode() Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. import java.util.HashMap;
2. public class HashDemo
3. {
4.     public static void main(String a[])
5.     {
6.         HashMap<Price, String> hm = new HashMap<Price, String>();
7.         hm.put(new Price("Banana", 20), "Banana");
8.         hm.put(new Price("Apple", 40), "Apple");
9.         hm.put(new Price("Orange", 30), "Orange"); //creating new object to use as key to get value
10.        Price key = new Price("Banana", 20);
11.        System.out.println("Hashcode of the key: "+key.hashCode());
12.        System.out.println("Value from map: "+hm.get(key));
13.    }
14.}
15.class Price
16.
17. private String item;
18. private int price;
19. public Price(String itm, int pr)
20.
21. {
22.     this.item = itm;
23.     this.price = pr;
24. }
25. public int hashCode()
26.
27. {
28.     System.out.println("In hashCode");
29.     int hashCode = 0;
30.     hashCode = price*20;
31.     hashCode += item.hashCode();
32.     return hashCode;
33. }
34. public boolean equals(Object obj)
35.
36. {
37.     System.out.println("In equals");
38.     if(obj instanceof Price)
39.     {
40.         Price pp = (Price) obj;
41.         return (pp.item.equals(this.item) && pp.price == this.price);
42.     }
43.     else
44.     {
45.         return false;
46.     }
47. }
```

```
45. public String getItem()
46. {
47.     return item;
48. }
49. public void setItem(String item)
50. {
51.     this.item = item;
52. }
53. public int getPrice()
54. {
55.     return price;
56. }
57. public void setPrice(int price)
58. {
59.     this.price = price;
60. }
61. public String toString()
62. {
63.     return "item: "+item+" price: "+price;
64. }
65. }
```

Output:

```
$ javac HashDemo.java
$ java HashDemo
```

```
In hashCode
In hashCode
In hashCode
In hashCode
In hashCode
HashCode of the key: 1982479637
In hashCode
In equals
Value from map: Banana
```

122. Java Program to Demonstrate Usage of an Instance Variable x in the Test Class

[« Prev](#)

[Next »](#)

This is a Java Program to Demonstrate Usage of an Instance Variable x in the Test Class.

Here we made a variable X as an instance variable and if we try to access that from main() directly then it will show error and it proves that all non – static things of a class always belong to an object. Thus, keep those properties of an object as an instance whose values are changing from each instance of an object.

Here is the source code of the Java Program to Demonstrate Usage of an Instance Variable x in the Test Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Test
2. {
3.     int x = 17;
4.     public static void main(String...a)
5.     {
6.         //System.out.println(x); Error : non-static variable x cannot be referenced from a static context
7.         // non static variable can be called only after making objects
8.         Test ob = new Test();
9.         System.out.println(ob.x);
10.    }
11. }
```

Output:

```
$ javac Test.java
$ java Test
```

17

123. Java Program to Demonstrate Usage of a Static Variable x in the Test Class

[« Prev](#)

[Next »](#)

This is a Java Program to Demonstrate Usage of a Static Variable x in the Test Class.

Here we made a variable X as a static variable and if we try to access that from main() directly then it will not show any error and executed. It proves that all static things of a class always belong to a class. Thus, keep those properties of an object as a static whose values are same from each instance of an object.

Here is the source code of the Java Program to Demonstrate Usage of a Static Variable x in the Test Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Test
2. {
3.     static int x = 17;
4.     public static void main(String...a)
5.     {
6.         System.out.println(x);           //static variable can be called directly
7.         System.out.println(Test.x);   //static variable can be called with class its name
8.     }
9. }
```

Output:

```
$ javac Test.java
$ java Test
```

```
17
17
```

124. Java Program to Illustrates Use of Chaining Constructor

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrates Use of Chaining Constructor.

Here, we use this() to achieve Constructor Chaining and there must be atleast one constructor which is not having this() as a first line. Constructor Chaining can be achieved in any order.

Here is the source code of the Java Program to Illustrates Use of Chaining Constructor. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class ConstructorChaining
2. {
3.     ConstructorChaining()
4.     {
5.         this(20);
6.         System.out.println("Default constructor of class.");
7.     }
8.     ConstructorChaining(int x)
9.     {
10.        System.out.println("Parameterized (1 parameter) constructor of class.");
11.        System.out.println("The value of x is "+x);
12.    }
13.    ConstructorChaining(int x, int y)
14.    {
15.        this();
16.        System.out.println("Parameterized (2 parameters) constructor of class.");
17.        System.out.println("The value of x and y is " + x + "and " + y + ". " + "The sum of x and y is " + (x + y));
18.    }
19.    public static void main(String... a)
20.    {
21.        ConstructorChaining(11,12);
22.    }
23.}
```

Output:

```
$ javac ConstructorChaining.java
$ java ConstructorChaining
```

```
Parameterized (1 parameter) constructor of class.
The value of x is 20
Default constructor of class.
Parameterized (2 parameters) constructor of class.
The value of x and y is 11and 12. The sum of x and y is 23
```

125. Java Program to Illustrates Use of Abstract Class and Method

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrates Use of Abstract Class and Method.

Here we made add() and multiply() as abstract methods in parent class which are defined in child class and we can call all these methods via making object of child class.

Here is the source code of the Java Program to Illustrates Use of Abstract Class and Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. abstract class Calculation
2. {
3.     float a = 12, b = 6, c;
4.     abstract void add();
5.     void subtract()
6.     {
7.         c = a - b;
8.         System.out.println("Result:"+c);
9.     }
10.    abstract void multiply();
11.    void divide()
12.    {
13.        c = a / b;
14.        System.out.println("Result:"+c);
15.    }
16.
17.public class AbstractionDemo extends Calculation
18.
19.    void add()
20.    {
21.        c = a + b;
22.        System.out.println("Result:"+c);
23.    }
24.    void multiply()
25.    {
26.        c = a * b;
27.        System.out.println("Result:"+c);
28.    }
29.    public static void main(String[] args)
30.    {
31.        AbstractionDemo obj = new AbstractionDemo;
32.        obj.add();
33.        obj.subtract();
34.        obj.multiply();
35.        obj.divide();
36.    }
37.}
```

Output:

```
$ javac AbstractionDemo.java
$ java AbstractionDemo
```

```
Result:18.0
Result:6.0
Result:72.0
Result:2.0
```

126. Java Program to Illustrates Use of Static Inner Class

[« Prev](#)

[Next »](#)

This is a Java Program to Illustrates Use of Static Inner Class.

We made a Outer class and define a static Inner class in it. We define a non static method as show() method in inner class and we try to call this method by inner class object and outer class object and we get the desired output.

Here is the source code of the Java Program to Illustrates Use of Static Inner Class. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1. public class Outer           //outer class
2. {
3.     int x = 10;
4.     static int y = 20;
5.     static class Inner        //static inner class
6.     {
7.         void show()          // non static method in static inner class
8.         {
9.             System.out.println(y);
10.            // System.out.println(x); error : non-static variable x cannot be referenced from a static context
11.        }
12.    }
13.    public static void main(String... s)
14.    {
15.        System.out.println("In main method'. The value of static data member of outer class is :" +y);
16.        // System.out.println(x);           error: non-static variable x cannot be referenced from a static context
17.        System.out.println("Inner class method accessed by Inner class object ");
18.        Inner i = new Inner();
19.        i.show();
20.        System.out.println("Inner class method accessed by outer class object ");
21.        Outer.Inner o = new Outer.Inner();
22.        o.show();
23.    }
24.}
```

Output:

```
$ javac Outer.java
$ java Outer
```

```
'In main method'. The value of static data member of outer class is :20
Inner class method accessed by Inner class object
20
Inner class method accessed by outer class object
20
```

127. Java Program to Handle MouseEvent

[« Prev](#)

[Next »](#)

This is a Java program to handle MouseEvent.

Problem Description

We have to write a program in Java such that it demonstrates the event actions associated with a mouse. The program should demonstrate various mouse events such as mouse clicked event, mouse pressed event, mouse released event, mouse entered event and mouse exited event.

Expected Input and Output

For handling MouseEvent, we can have the following 5 different sets of input and output.

advertisement

1. To test mouseClicked: When the mouse is clicked at a point on frame.

For example:

If the left button of the mouse is clicked at any point (x,y) on the frame
then the expected output is "Left Button Clicked at point x y"

2. To test mousePressed: When the mouse is pressed at a point on frame.

advertisement

For example:

If the left button of the mouse is kept pressed at any point (x,y) on the frame
then the expected output is "Left Button Pressed at point x y"

3. To test mouseReleased: When the mouse is released at a point on frame after being kept pressed.

For example:

advertisement

If the left button of the mouse is released at any point (x,y) on the frame
then the expected output is "Left Button Released at point x y"

4. To test mouseEntered: When the mouse enters the frame.

For example:

If the mouse enters into the frame from any point (x,y) on the frame
then the expected output is "Mouse Entered the frame from point x y"

5. To test mouseExited: When the mouse leaves the frame.

advertisement

For example:

If the mouse leaves the frame from any point (x,y) on the frame
then the expected output is "Mouse Exited the frame from point x y"

Problem Solution

1. The program uses the interfaces **MouseListener** and **ActionListener** of the **java.awt** package.

a) The **MouseListener** interface has five member methods :

i) **public void mouseEntered(MouseEvent):** This method gives the co-ordinates of the point from where the mouse has entered the frame.

ii) **public void mouseExited(MouseEvent):** This method gives the co-ordinates of the point from where the

mouse has left the frame.

iii) **public void mouseReleased(MouseEvent):** This method gives the co-ordinates of the point till where the mouse cursor was dragged before release and also the button clicked on the mouse.

iv) **public void mousePressed(MouseEvent):** This method gives the co-ordinates of the point from where the mouse was pressed and also the button.

v) **public void mouseClicked(MouseEvent):** This method gives the co-ordinates of the point where the mouse cursor was clicked and also the button clicked.

b) The **ActionListener** interface has member method :

public void actionPerformed(ActionEvent): This method works on the click of the exit button and the functions closes the frame.

2. **@Override** is a keyword used to override any method on the parent class. When the sub class has any method defined in parent/super class with same name and parameters, the Override keyword is used. This overrides the function of super class and executed the function defined in the sub class.

advertisement

3. Create a class that implements the two required interfaces – **MouseListener** and **ActionListener**.

4. Create a frame, text field and an exit button with required dimensions. After positioning the co-ordinates of the text field and button, add them to the frame.

5. Associate MouseListener with the frame and ActionListener with the exit button. Display the frame.

6. The functions **getX()** and **getY()** gives the x & y co-ordinates of the cursor respectively. Use them to display the necessary message in the text box.

Program/Source Code

Here is source code of the Java Program to handle the mouse events. The program is successfully compiled and tested using BlueJ on Windows 10 and javac compiler on Fedora 30.

```
1. /* Java Program to demonstrate the event actions associated with a mouse */
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. public class Mouse_Event implements MouseListener,ActionListener
6. {
7.     static JFrame frame;
8.     static JTextField text;
9.     //Driver function
10.    public static void main(String[] args)
11.    {
12.        //Create a Frame
13.        frame=new JFrame("Mouse Event");
14.        frame.setBackground(Color.white);
15.        frame.setSize(500,500);
16.        frame.setLayout(null);
17.        //Create a TextField
18.        text=new JTextField();
19.        text.setBounds(0,0,500,50);
20.        frame.add(text);
21.        //Create a exit button to close the frame
22.        JButton exit=new JButton("Exit");
23.        exit.setBounds(220,235,60,30);
24.        frame.add(exit);
25.        //Create an object of the class Mouse_Event
26.        Mouse_Event obj=new Mouse_Event();
27.        //Associate MouseListener with the frame
28.        frame.addMouseListener(obj);
29.        //Associate ActionListener with button exit
30.        exit.addActionListener(obj);
31.        //Display frame
32.        frame.setVisible(true);
33.    }
34.    //function to dispose the frame on click of exit button
35.    @Override
36.    public void actionPerformed(ActionEvent e)
37.    {
```

```

38.     frame.dispose();
39. }
40. //function to get co-ordinates from where cursor entered the frame
41. @Override
42. public void mouseEntered(MouseEvent e)
43. {
44.     text.setText("");
45.     text.setText("Mouse Entered the frame from point ");
46.     text.setText(text.getText()+e.getX()+" "+e.getY());
47. }
48. //function to get co-ordinates from where cursor exited the frame
49. @Override
50. public void mouseExited(MouseEvent e)
51. {
52.     text.setText("");
53.     text.setText("Mouse Exited the frame from point ");
54.     text.setText(text.getText()+e.getX()+" "+e.getY());
55. }
56. //function to get co-ordinates where mouse was released
57. @Override
58. public void mouseReleased(MouseEvent e)
59. {
60.     text.setText("");
61.     String button="Right";
62.     if(e.getButton()==MouseEvent.BUTTON1)
63.         button="Left";
64.     text.setText(button+" Button Released at point ");
65.     text.setText(text.getText()+e.getX()+" "+e.getY());
66. }
67. //function to get co-ordinates where and which button of mouse was pressed
68. @Override
69. public void mousePressed(MouseEvent e)
70. {
71.     text.setText("");
72.     String button="Right";
73.     if(e.getButton()==MouseEvent.BUTTON1)
74.         button="Left";
75.     text.setText(button+" Button Pressed at point ");
76.     text.setText(text.getText()+e.getX()+" "+e.getY());
77. }
78. //function to get co-ordinates where and which button of mouse was clicked
79. @Override
80. public void mouseClicked(MouseEvent e)
81. {
82.     text.setText("");
83.     String button="Right";
84.     if(e.getButton()==MouseEvent.BUTTON1)
85.         button="Left";
86.     text.setText(button+" Button Clicked at point ");
87.     text.setText(text.getText()+e.getX()+" "+e.getY());
88. }
89. }

```

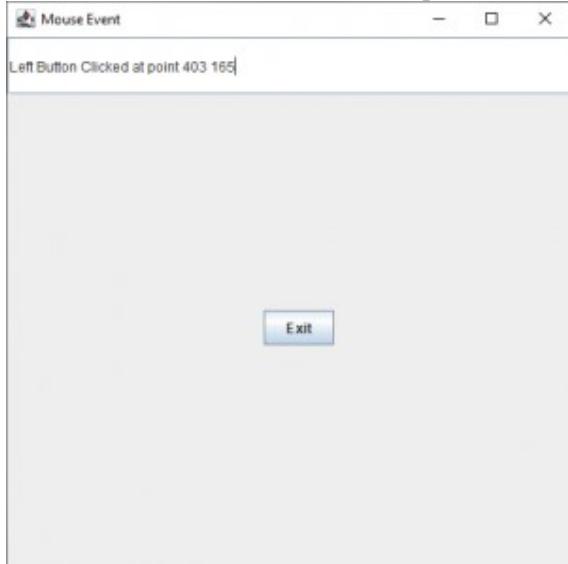
Program Explanation

1. The program demonstrates the functions of the **MouseListener** interface.
 - a) When the cursor enters the frame, the **mouseEntered** function is called.
 - b) When the cursor exits the frame, the **mouseExited** function is called.
 - c) When the cursor is released after being pressed, the **mouseReleased** function is called.
 - d) When the cursor is kept pressed at the frame, the **mousePressed** function is called.
 - e) When the cursor is clicked at a point on the frame, the **mouseClicked** function is called.
2. The **@Override** keyword overrides the functions of the parent class and executes the functions of sub class.

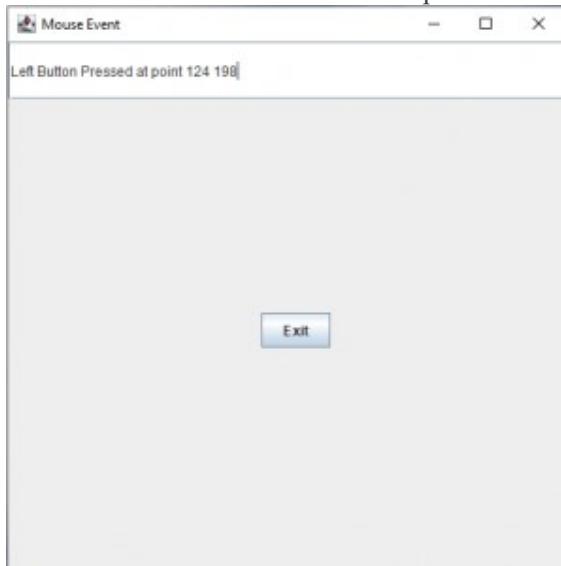
Runtime Test Cases

Here's the run time test cases for mouse events.

Test case 1 – Here's the runtime output of the button clicked operation.



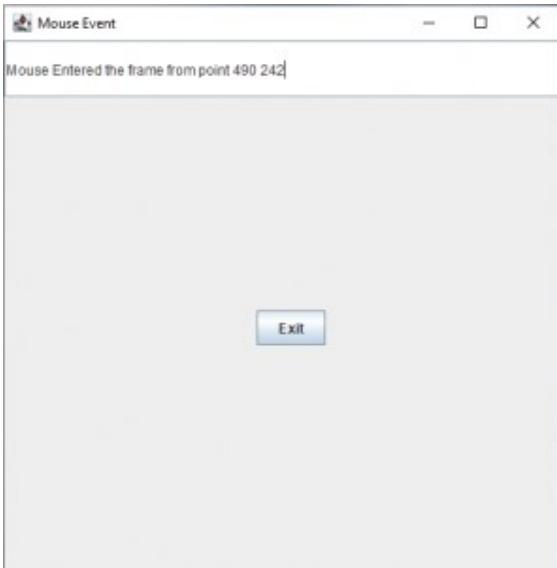
Test case 2 – Here's the runtime output of the button pressed operation.



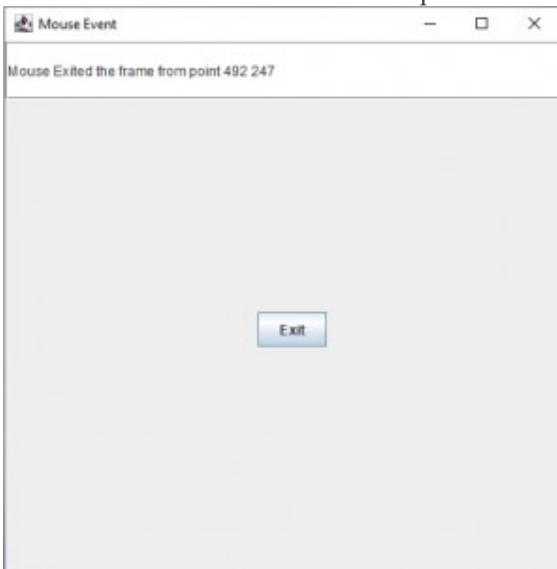
Test case 3 – Here's the runtime output of the button released operation.



Test case 4 – Here's the runtime output of the mouse entered operation.



Test case 5 – Here's the runtime output of the mouse exited operation.



128. Java Program to Handle KeyBoardEvent

[« Prev](#)

[Next »](#)

This is a Java program to handle KeyBoardEvent.

Problem Description

We have to write a program in Java such that it demonstrates the event actions associated with the keyboard. The program should demonstrate various keyboard events such as key typed event, key pressed event and key released event by using alphabets, digits and non aplha numeric keys.

Expected Input and Output

For handling KeyboardEvent, we can have the following 5 different sets of input and output.

advertisement

1. To test keyTyped: When a key representing an alphabet is stroked on the keyboard and it is the first entry of input field.

For example:

If the key 'h' is the stroked/typed key on the keyboard
then the expected output is "Key Typed : h"

2. To test keyTyped: When a key representing an alphabet is stroked on the keyboard after multiple entries in input field.

advertisement

For example:

If the key 'o' is the last stroked/typed key on the keyboard
then the expected output is "Key Typed : o"

3. To test keyTyped: When a key representing a digit is stroked on the keyboard after multiple entries in input field.

For example:

If the key '1' is the last stroked/typed key on the keyboard
then the expected output is "Key Typed : 1"

4. To test keyTyped: When a key representing a digit is stroked on the keyboard after multiple entries in input field.

For example:

advertisement

If the key '7' is the last stroked/typed key on the keyboard
then the expected output is "Key Typed : 7"

5. To test keyPressed: When a non aplha numeric key is kept pressed on the keyboard.

For example:

If the key 'alt' is the kept pressed on the keyboard
then the expected output is "Key Pressed : 18"
Note: 18 is the unicode of the alt key.

6. To test keyReleased: When a non aplha numeric key is pressed and released on the keyboard.

For example:

advertisement

If the key 'alt' is pressed and released on the keyboard
then the expected output is "Key Released : 18"
Note: 18 is the unicode of the alt key.

Problem Solution

1. The program uses the interfaces **KeyListener** and **ActionListener** of the **java.awt** package.
 - a) The **KeyListener** interface has three member methods :
 - i) **public void keyReleased(KeyEvent)**: This method gives the Unicode of the key released and its character equivalent if the key pressed is a letter.
 - ii) **public void keyPressed(KeyEvent)**: This method gives the Unicode of the key pressed and its character equivalent if the key pressed is a letter.
 - iii) **public void keyTyped(KeyEvent)**: This method gives the character equivalent of the key pressed provided it is a valid character.
 - b) The **ActionListener** interface has member method :
public void actionPerformed(ActionEvent): This method works on the click of the exit button and the functions closes the frame.
 2. **@Override** is a keyword used to override any method on the parent class. When the sub class has any method defined in parent/super class with same name and parameters, the Override keyword is used. This overrides the function of super class and executed the function defined in the sub class.
 3. Create a class that implements the two required interfaces – **KeyListener** and **ActionListener**.
 4. Create a frame, two text fields for input & output and an exit button with required dimensions. After positioning the co-ordinates of the text fields and button, add them to the frame.
 5. Associate **KeyListener** with the input text field and **ActionListener** with the exit button.
 6. Display the frame.
 7. The functions **getKeyCode()** and **getKeyChar()** give the unicode and character representation of the key pressed respectively. Use them to display the necessary message in the output text box.

advertisement

Program/Source Code

Here is source code of the Java Program to handle the keyboard events. The program is successfully compiled and tested using BlueJ on Windows 10 and javac compiler on Fedora 30.

```
1. /* Java Program to demonstrate the event actions associated with a keyboard */
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Keyboard_Event implements KeyListener,ActionListener
6. {
7.     static JFrame frame;
8.     static JTextField output;
9.     static JTextField input;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.        //Create a frame
14.        frame=new JFrame("Keyboard Event");
15.        frame.setBackground(Color.white);
16.        frame.setSize(500,500);
17.        frame.setLayout(null);
18.        //Create a text field for output
19.        output=new JTextField();
20.        output.setBounds(0,0,500,50);
21.        frame.add(output);
22.        //Create a text field for input
23.        input=new JTextField();
24.        input.setBounds(0,400,500,50);
25.        frame.add(input);
26.        //Create an exit button
27.        JButton exit=new JButton("Exit");
28.        exit.setBounds(220,200,60,30);
29.        frame.add(exit);
30.        //Create an object of the class
31.        Keyboard_Event obj=new Keyboard_Event();
32.        //Associate KeyListener with input
```

```

33.    input.addKeyListener(obj);
34.    //Associate ActionListener with exit
35.    exit.addActionListener(obj);
36.    frame.setVisible(true);
37. }
38. //function to dispose the frame when exit button is clicked
39. @Override
40. public void actionPerformed(ActionEvent e)
41. {
42.    frame.dispose();
43. }
44. /*function to display the unicode of key released
45. and the character if it is a letter or a digit*/
46. @Override
47. public void keyReleased(KeyEvent e)
48. {
49.    output.setText("");
50.    output.setText("Key Released : "+e.getKeyCode());
51.    if(Character.isLetter(e.getKeyChar()))
52.        keyTyped(e);
53.    if(Character.isDigit(e.getKeyChar()))
54.        keyTyped(e);
55. }
56. /*function to display the unicode of key pressed
57. and the character if it is a letter or a digit*/
58. @Override
59. public void keyPressed(KeyEvent e)
60. {
61.    output.setText("");
62.    output.setText("Key Pressed : "+e.getKeyCode());
63.    if(Character.isLetter(e.getKeyChar()))
64.        keyTyped(e);
65.    if(Character.isDigit(e.getKeyChar()))
66.        keyTyped(e);
67. }
68. //function to display the character of the key typed
69. @Override
70. public void keyTyped(KeyEvent e)
71. {
72.    output.setText("");
73.    output.setText("Key Typed : "+e.getKeyChar());
74. }
75.

```

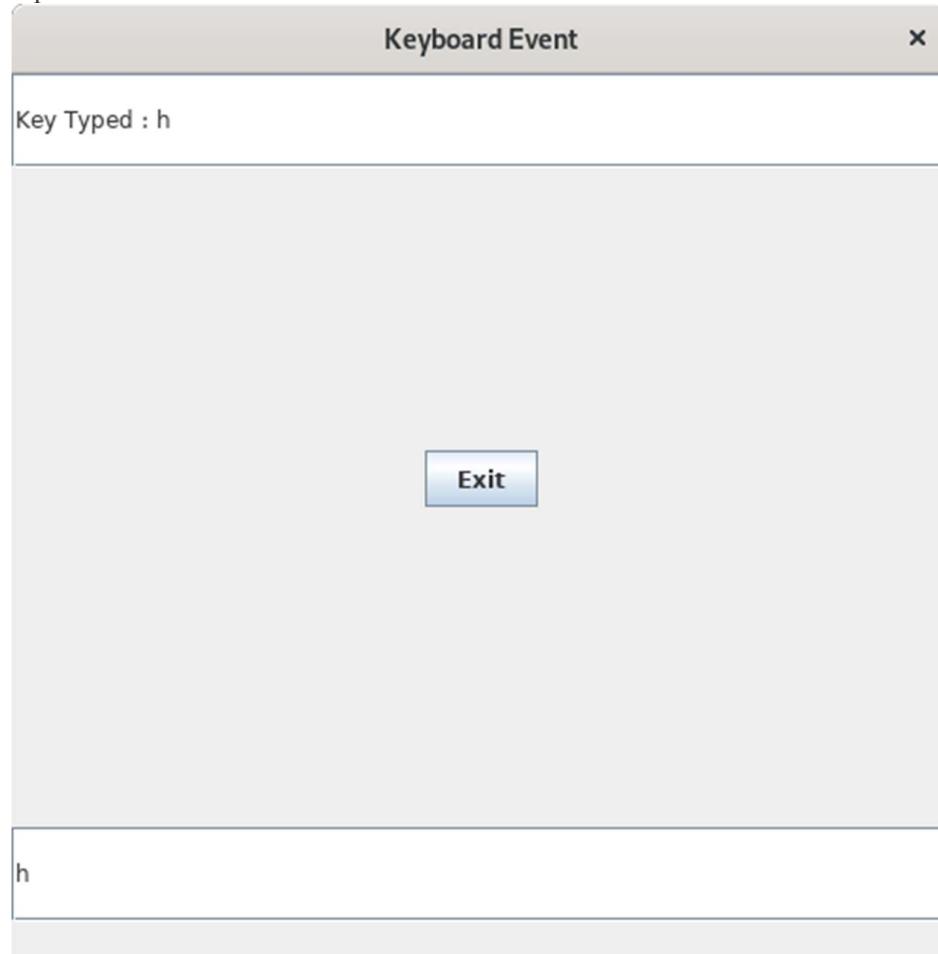
Program Explanation

1. **Input** and **Output** are the two text fields for input of string and output of the key action respectively.
2. The program demonstrates the functions of the **KeyListener** interface.
 - a) When the key is released, the **keyReleased** function is called which displays the unicode of the key and if the key is a valid letter then calls the method **keyTyped**.
 - b) When the key is pressed, the **keyPressed** function is called which displays the unicode of the key and if the key is a valid letter then calls the method **keyTyped**.
 - c) When the key is typed, the **keyTyped** function is called which displays the character representation of the key pressed.
3. The **@Override** keyword overrides the functions of the parent class and executes the functions of sub class.

Runtime Test Cases

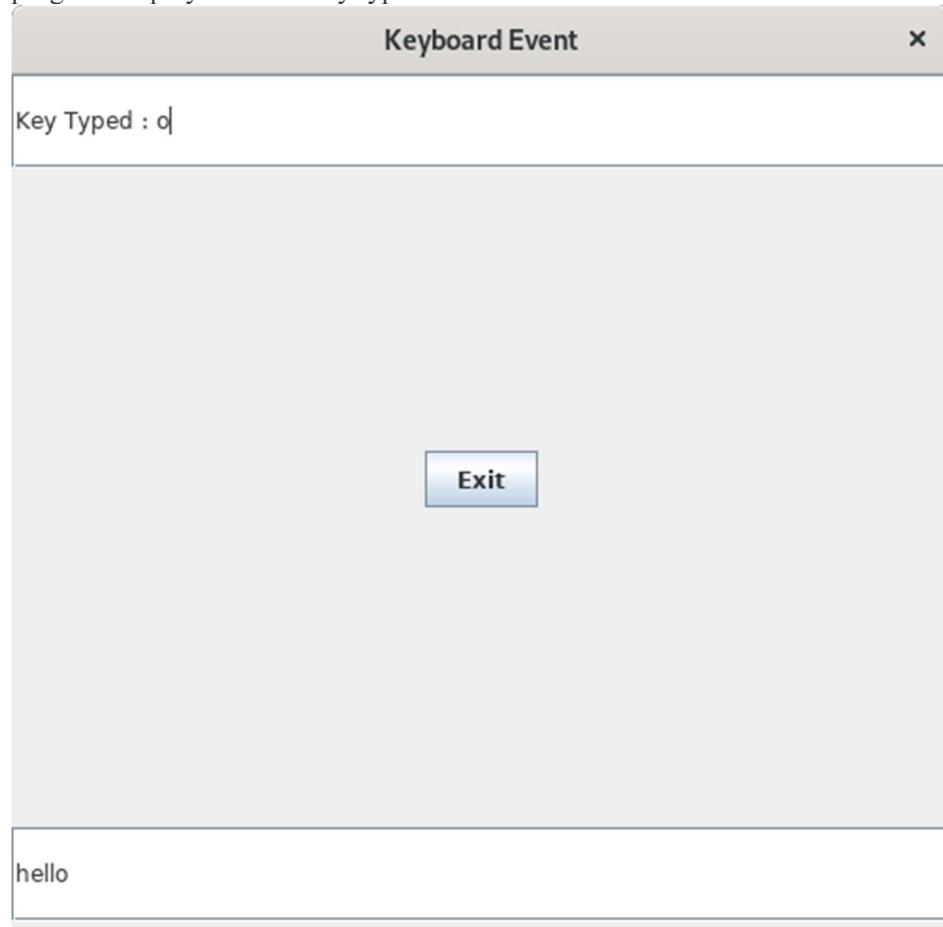
Here's the run time test cases for keyboard events.

Test case 1 – Here's the runtime output of the key typed operation with key “h”, where “h” is first entry of the input field.



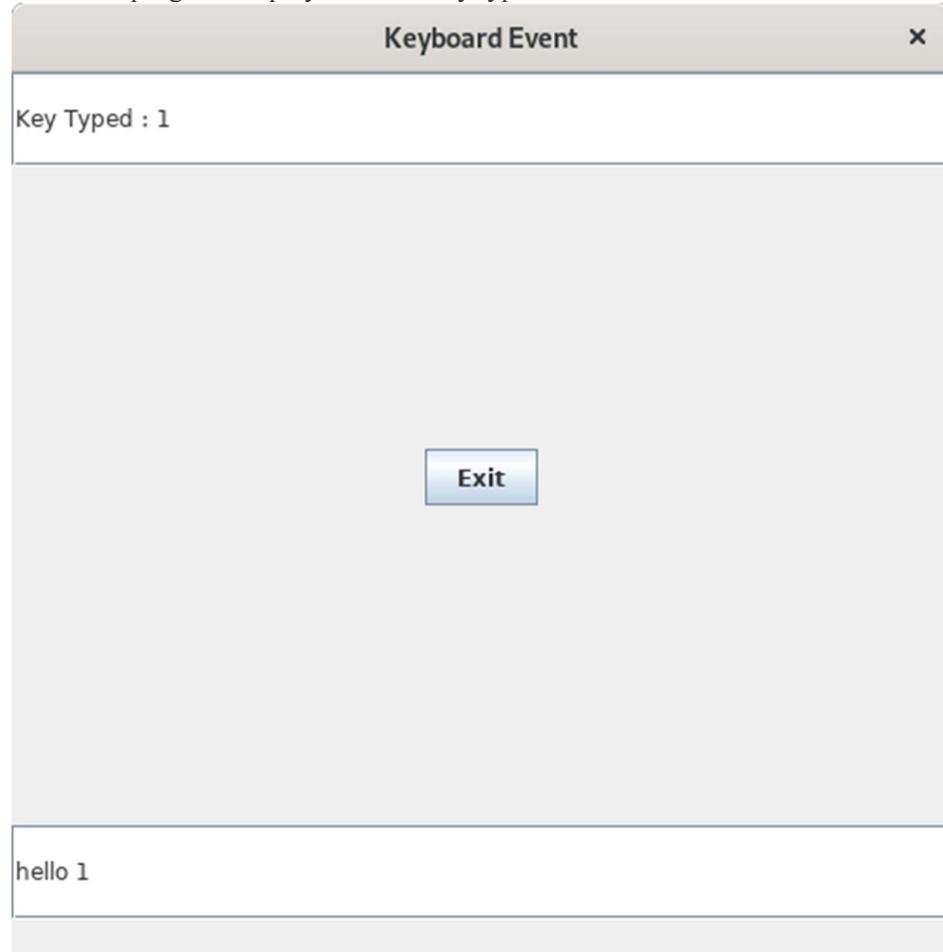
Test case 2 – Here's the runtime output of the multiple entries in key typed operation by using alphabets in the input field. For example, here the entered key is “hello”, where “o” is the last entry of the input field. The

program displays that the key typed is o.



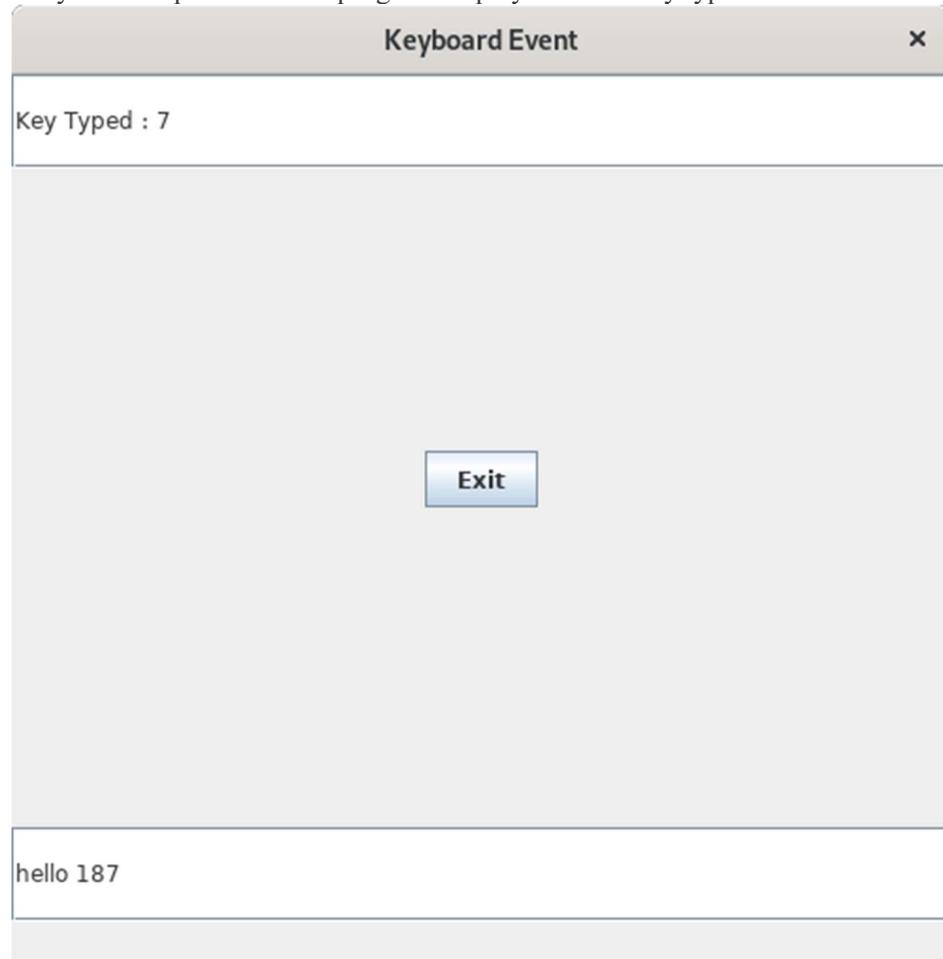
Test case 3 – Here's the runtime output of the multiple entries in key typed operation by using alphabets and digits in the input field. For example, here the entered key is “hello 1”, where “1” is the last entry of the input

field. The program displays that the key typed is 1.

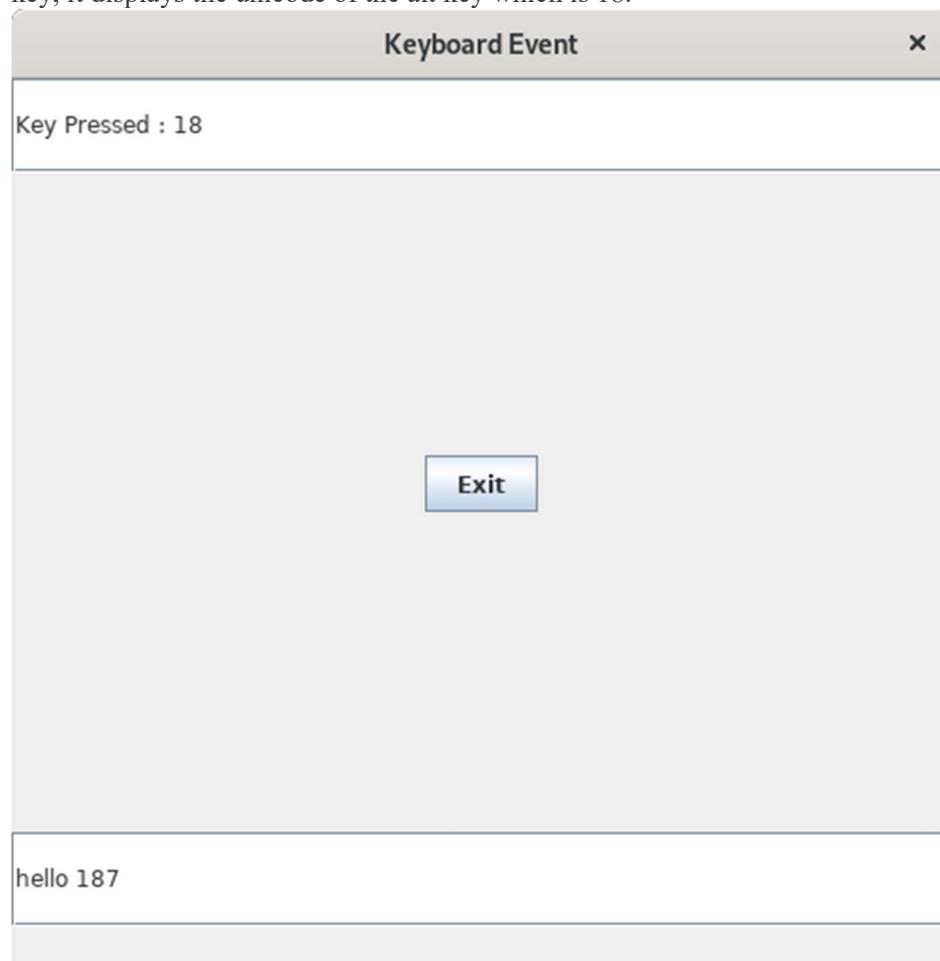


Test case 4 – Here's the runtime output of the multiple entries in key typed sequence operation by using alphabets and digits in the input field. For example, here the entered key is “hello 187”, where “7” is the last

entry of the input field. The program displays that the key typed is 7.



Test case 5 – Here's the runtime output of the special key pressed operation with key “alt”. After pressing alt key, it displays the unicode of the alt key which is 18.



129. Java Program to Close the Frame Using WindowAdapter Class

[« Prev](#)

[Next »](#)

This is a Java program to demonstrate the WindowAdapter class to close the frame.

Problem Description

We have to write a program in Java such that it demonstrates the use of the **WindowAdapter** class to close the frame. The program should demonstrate various WindowAdapter class functions such as window opened function, window activated function, window deactivated function, window iconified function, window deiconified function and window closing function.

The **WindowAdapter** class is a class which implements the **WindowListener** interface.

advertisement

Expected Input and Output

For WindowEvent class, we can have the following 6 different sets of input and output.

1. To test windowOpened: When the frame is opened after creation.

For example:

If the frame is created, that is when the program is first executed
then the expected output is

"Status of Frame : Activated
Status of Frame : Opened"

2. To test windowDeactivated: When the frame is deactivated.

advertisement

For example:

If the frame is deactivated
then the expected output is "Status of Frame : Deactivated"

3. To test windowActivated: When the frame is activated.

advertisement

For example:

If the frame is activated
then the expected output is "Status of Frame : Activated"

4. To test windowIconified: When the frame is iconified.

For example:

If the frame is iconified, that is the frame is minimized
then the expected output is
"Status of Frame : Iconified
Status of Frame : Deactivated"

5. To test windowDeiconified: When the frame is deiconified.

advertisement

For example:

If the frame is deiconified, that is the frame is maximized
then the expected output is
"Status of Frame : Deiconified"

Status of Frame : Activated"

6. To test windowClosing: When the frame is to be closed.

For example:

If the frame is being closed,

then the expected output is

"Status of Frame : Closing

Status of Frame : Deactivated"

Problem Solution

1. The program uses the **WindowAdapter** class to close the frame. The **WindowListener** interface has seven member methods :

i) **public void windowClosing(WindowEvent):** This method prints the status of frame as “Closing”, when the frame is being closed.

ii) **public void windowClosed(WindowEvent):** This method closes the frame.

iii) **public void windowIconified(WindowEvent):** This method prints the status of frame as “Iconified”, when the frame in minimized.

iv) **public void windowDeiconified(WindowEvent):** This method prints the status of frame as “Deiconified”, when the frame in maximized.

v) **public void windowActivated(WindowEvent):** This method prints the status of frame as “Activated”, when the frame is active.

vi) **public void windowDeactivated(WindowEvent):** This method prints the status of frame as “Deactivated”, when the frame is not active.

vii) **public void windowOpened(WindowEvent):** This methods prints the status of frame as “Opened”, when the frame is displayed after creation.

advertisement

2. Create a class that inherits the **WindowAdapter** class and implements the **WindowListener** interface.

3. Create a frame and associate the **WindowListener** interface of **WindowAdapter** class with the frame.

4. Use the functions of **WindowListener** interface to print the necessary message.

Program/Source Code

Here is source code of the Java Program to handle the keyboard events. The program is successfully compiled and tested using BlueJ on Windows 10 and javac compiler on Fedora 30.

```
1. /* Java Program to demonstrate the WindowAdapter Class */
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Window_Adapter extends WindowAdapter
6. {
7.     static JFrame frame;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        frame=new JFrame("Window Adapter Class");
13.        frame.setBackground(Color.white);
14.        frame.setSize(500,500);
15.        //Create an object of the class
16.        Window_Adapter obj=new Window_Adapter();
17.        //Associate WindowListener with frame
18.        frame.addWindowListener(obj);
19.        frame.setVisible(true);
20.    }
21.    //function to display status as closing when the frame is closed
22.    @Override
23.    public void windowClosing(WindowEvent e)
24.    {
25.        System.out.println("Status of frame : Closing");
26.        windowClosed(e);
27.    }
```

```

28. //function to close the frame
29. @Override
30. public void windowClosed(WindowEvent e)
31. {
32.     frame.dispose();
33. }
34. //function to display status as iconified when the frame is minimized
35. @Override
36. public void windowIconified(WindowEvent e)
37. {
38.     System.out.println("Status of frame : Iconified");
39. }
40. //function to display status as deiconified when the frame is maximized
41. @Override
42. public void windowDeiconified(WindowEvent e)
43. {
44.     System.out.println("Status of frame : Deiconfied");
45. }
46. //function to display status as activated when the frame is active
47. @Override
48. public void windowActivated(WindowEvent e)
49. {
50.     System.out.println("Status of frame : Activated");
51. }
52. //function to display status as deactivated when the frame is inactive
53. @Override
54. public void windowDeactivated(WindowEvent e)
55. {
56.     System.out.println("Status of frame : Deactivated");
57. }
58. //function to display status as opened when the frame is created
59. @Override
60. public void windowOpened(WindowEvent e)
61. {
62.     System.out.println("Status of frame : Opened");
63. }
64.

```

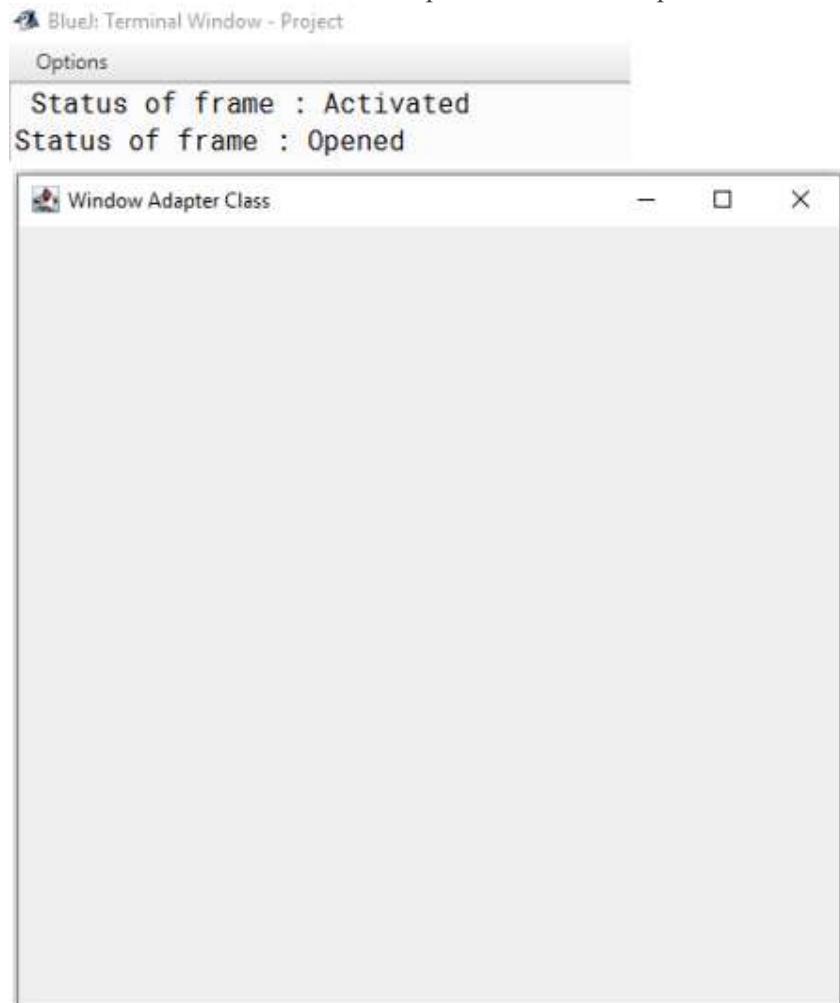
Program Explanation

1. This program uses the **WindowAdapter** class to inherit the features of **WindowListener** interface.
2. A frame is created and **WindowListener** is associated with the frame.
3. The suitable functions are called by the interface depending on the activities of the frame.
 - a) When the frame is created, the **windowOpened** function is called.
 - b) When the frame is deactivated, the **windowDeactivated** function is called.
 - c) When the frame is activated, the **windowActivated** function is called.
 - d) When the frame is de-iconified, the **windowDeiconified** function is called.
 - e) When the frame is iconified, the **windowIconified** function is called.
 - f) When the frame is closing, the **windowClosing** function is called which further calls the **windowClosed** function.
 - g) The **windowClosed** function disposes off the frame.

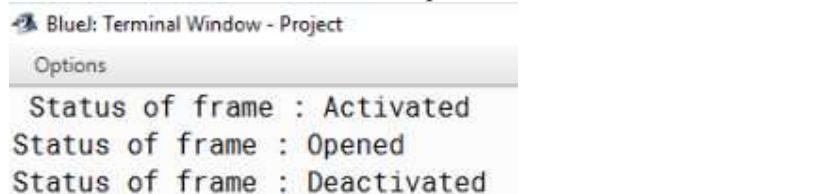
Runtime Test Cases

Here's the run time test cases for WindowAdapter Class.

Test case 1 – Here's the runtime output of the windowOpened function.



Test case 2 – Here's the runtime output of the windowDeactivated function.



Test case 3 – Here's the runtime output of the windowActivated function.

```
Blue: Terminal Window - Project
Options
Status of frame : Activated
Status of frame : Opened
Status of frame : Deactivated
Status of frame : Activated
```

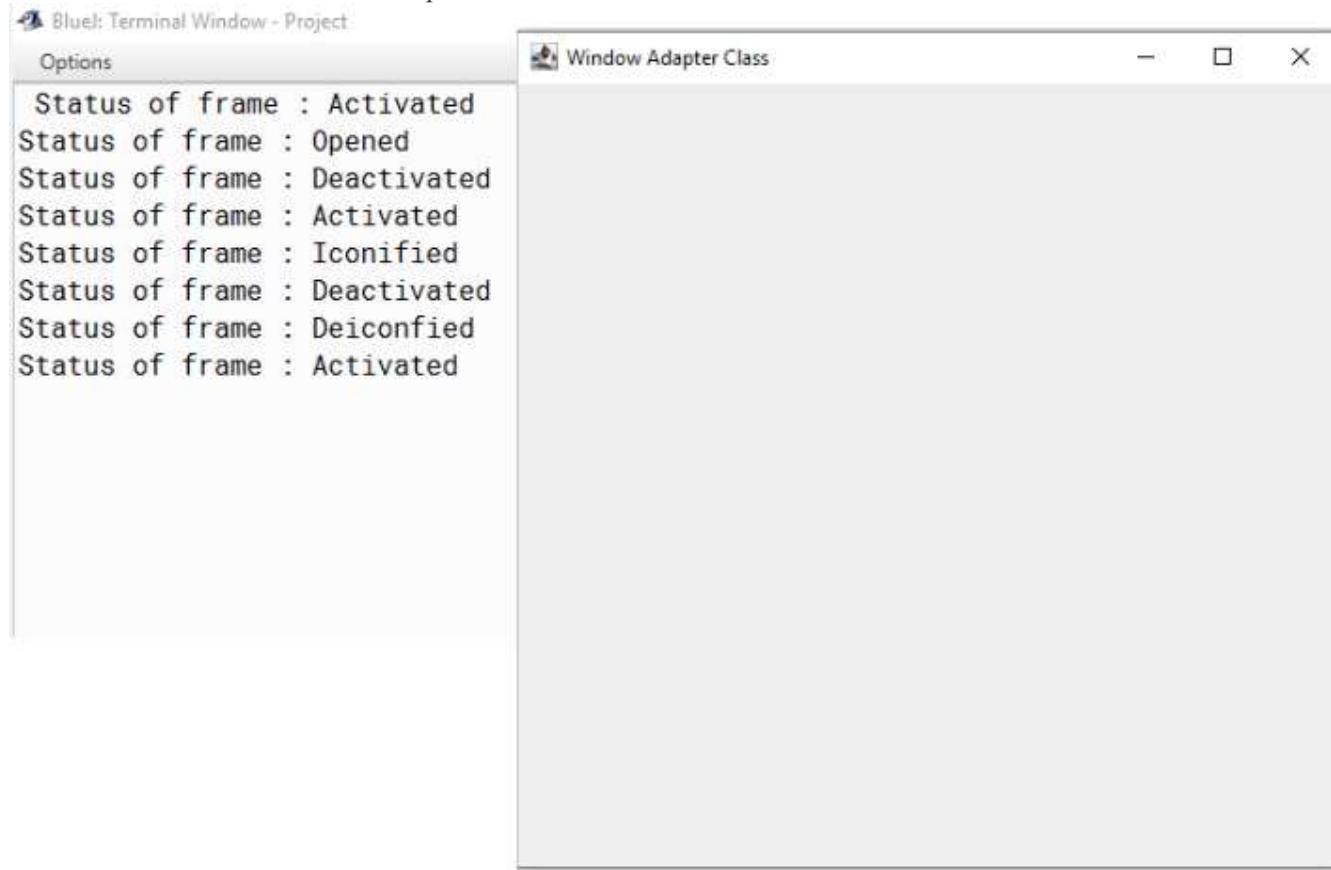


The screenshot shows a terminal window titled "Window Adapter Class". The window has standard minimize, maximize, and close buttons at the top right. Below the title bar, there is a single button labeled "Minimize". The main area of the terminal displays the runtime output of the windowActivated function, which consists of four lines of text: "Status of frame : Activated", "Status of frame : Opened", "Status of frame : Deactivated", and "Status of frame : Activated".

Test case 4 – Here's the runtime output of the windowIconified function.

```
Blue: Terminal Window - Project
Options
Status of frame : Activated
Status of frame : Opened
Status of frame : Deactivated
Status of frame : Activated
Status of frame : Iconified
Status of frame : Deactivated
```

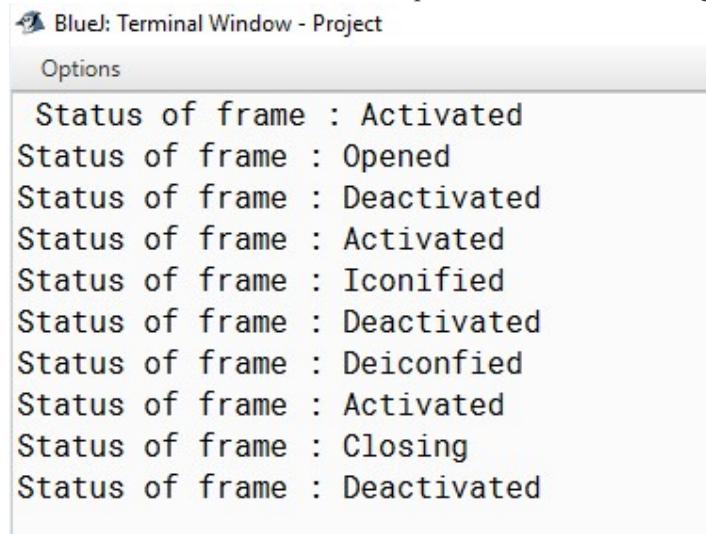
Test case 5 – Here's the runtime output of the windowDeiconified function.



A screenshot of a BlueJ IDE terminal window titled "BlueJ: Terminal Window - Project". The window contains the following text output:

```
Status of frame : Activated
Status of frame : Opened
Status of frame : Deactivated
Status of frame : Activated
Status of frame : Iconified
Status of frame : Deactivated
Status of frame : Deiconfied
Status of frame : Activated
```

Test case 6 -Here's the runtime output of the windowClosing function.



A screenshot of a BlueJ IDE terminal window titled "BlueJ: Terminal Window - Project". The window contains the following text output:

```
Status of frame : Activated
Status of frame : Opened
Status of frame : Deactivated
Status of frame : Activated
Status of frame : Iconified
Status of frame : Deactivated
Status of frame : Deiconfied
Status of frame : Activated
Status of frame : Closing
Status of frame : Deactivated
```

130. Java Program to Close the Frame Using an Anonymous Class

[« Prev](#)

[Next »](#)

This is a Java program to close the frame using an anonymous class.

Problem Description

We have to write a program in Java such that it closes a frame on click of an exit button using an anonymous class.

Expected Input and Output

For closing of frame using an anonymous class, we can have the following 2 different sets of input and output.

advertisement

1. To test the opening of the frame: When the frame is opened after creation.

For example:

If the frame is created and opening, that is when the program is first executed
then the expected output is "Frame Opened Successfully"

2. To test the closing of the frame: When the frame is closed.

advertisement

For example:

If the frame is closed, that is on click of the "Exit" button on frame
then the expected output is "Frame Closed Successfully"

Problem Solution

1. Create a frame, an exit button and object of the class.
2. Define a constructor of the class which checks the actions on the exit button and disposes the frame if the button is clicked.

advertisement

Program/Source Code

Here is source code of the Java Program to handle the keyboard events. The program is successfully compiled and tested using BlueJ on Windows 10 and javac compiler on Fedora 30.

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4. class Close_Frame
5. {
6.     static JFrame frame;
7.     static JButton exit;
8.     //Constructor
9.     Close_Frame()
10.    {
11.        exit.addActionListener(new ActionListener()
12.        {
13.            @Override
14.            public void actionPerformed(ActionEvent e)
15.            {
```

```

15.     frame.dispose();
16.     System.out.println("Frame Closed Successfully");
17.   }
18. }
19. }
20. //Driver Function
21. public static void main(String args[])
22. {
23.   //Create a frame
24.   frame=new JFrame("Close Frame Using Anonymous Class");
25.   frame.setSize(500,500);
26.   frame.setBackground(Color.white);
27.   frame.setLayout(null);
28.   //Create an exit button
29.   exit=new JButton("Exit");
30.   exit.setBounds(220,235,60,30);
31.   frame.add(exit);
32.   //Create an object of the class
33.   Close_Frame obj=new Close_Frame();
34.   //Display frame
35.   frame.setVisible(true);
36.   System.out.println("Frame Opened Successfully");
37. }
38.

```

Program Explanation

1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.
2. Create an exit button with **setBounds(int x, int y, int width, int height)**, where ‘x’ & ‘y’ is used to set the position of top-left corner. ‘int width’ & ‘int height’ is used to set the width and height of the exit button.
3. Here the values of set bounds are **setBounds(220,235,60,30)**, where ‘220’ & ‘235’ is used to set the position of button from top and left side of the frame. ‘60’ & ‘30’ is used to set the width and height of the exit button.
4. Create an object of the class with a name **Close_Frame**. When the object is created, the constructor is called and it checks the event of the button.
5. When the exit button is clicked, the constructor closes the frame and displays suitable message.

advertisement

Runtime Test Cases

Here's the run time test cases for closing of frame using an anonymous class.

Test case 1 – Here's the runtime output of the frame opening. The program will create and displays the frame.

BlueJ: Terminal Window - Project

Options

Frame Opened Successfully

 Close Frame Using Anonymous Class

—

□

×

Exit

Test case 2 – Here's the runtime output of the frame closing. When the exit button is clicked, it closes the frame and displays suitable message.

BlueJ: Terminal Window - Project

Options

Frame Opened Successfully

Frame Closed Successfully

131. Java Program to Draw a Smiling Face Using the Methods of Graphics Class

[« Prev](#)

[Next »](#)

This is a Java program to draw a smiling face using graphics class.

Problem Description

We have to write a program in Java such that it creates a frame containing a smiling face using **Graphics** class. The Graphics class defines a number of drawing methods like **drawLine**, **drawOval**, **drawRect**, **drawArc**, **fillArc**, **fillOval**, **fillRect**, and many others.

advertisement

The default color for Graphics class methods is black. To change the color method **setColor** is used.

Expected Input and Output

For drawing a Smiling Face using Graphics class, we can have the following set of input and output.

To draw a Smiling Face:

We expect a Smiling Face with background color of the face as yellow. The eyes are to be properly positioned and filled with color black. The smile is to be drawn with black color with proper positioning.

advertisement

Problem Solution

1. Create a class that inherits the **JPanel** class.
2. To draw the smiling face, define the function : **public void paint(Graphics)**. This function is invoked to draw the shapes as defined in the function body.
3. Create three circles using **drawArc** for the face and two eyes and fill them with color of your choice.
4. Create an arc using **drawArc** for the smile with the color of your choice.

Program/Source Code

Here is source code of the Java Program to draw a simling face using Graphics class. The program is successfully compiled and tested using BlueJ on Windows 10 and javac compiler on Fedora 30.

```
1. import javax.swing.*;
2. import java.awt.*;
3. class Smiley extends JPanel
4. {
5.     //Driver function
6.     public static void main(String args[])
7.     {
8.         //Create a frame
9.         JFrame frame=new JFrame("Smiley");
10.        frame.setSize(500,500);
11.        frame.setBackground(Color.white);
12.        Smiley panel=new Smiley();
13.        frame.add(panel);
14.        //Set default close operation for the frame
15.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.        frame.setVisible(true);
17.    }
18.    //function to draw the shapes
19.    public void paint(Graphics g)
20.    {
21.        //Change color to yellow
22.        g.setColor(Color.yellow);
23.        //Draw and fill the face
24.        g.drawArc(100,100,250,250,0,360);
```

```

25.     g.fillArc(100,100,250,250,0,360);
26.     //Change color to black
27.     g.setColor(Color.black);
28.     //Draw the left eye
29.     g.drawArc(170,185,25,25,0,360);
30.     g.fillArc(170,185,25,25,0,360);
31.     //Draw the right eye
32.     g.drawArc(255,185,25,25,0,360);
33.     g.fillArc(255,185,25,25,0,360);
34.     //Draw the smile
35.     g.drawArc(150,215,150,100,0,-180);
36.   }
37.

```

Program Explanation

1. Create a class **Smiley** that inherits the **JPanel** class.
2. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.
3. Create an panel of the class **Smiley panel=new Smiley();** and add it to the frame by using **frame.add(panel);**
4. **frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);** is a default close operation for the frame.
5. **frame.setVisible(true);** is used to display the frame.
6. **public void paint(Graphics g)** this function is used to draw the smiling face. This function is invoked to draw the shapes as defined in the function body.
7. The **drawArc**, **fillArc** and **setColor** functions are defined in the **java.awt.Graphics** package.
 - a) **drawArc(int x,int y,int wid,int len,int start,int end):** This method draws an arc of length ‘len’ and width ‘wid’ starting from ‘x’ & ‘y’ co-ordinate from the angle ‘start’ to angle ‘end’.
g.drawArc(100,100,250,250,0,360); – This method is used to draw the circle face.
g.drawArc(170,185,25,25,0,360); – This method is used to draw the left eye.
g.drawArc(255,185,25,25,0,360); – This method is used to draw the right eye.
g.drawArc(150,215,150,100,0,-180); – This method is used to draw the smile.
 - b) **fillArc(int x,int y,int wid,int len,int start,int end):** This method fills the arc of length ‘len’ and width ‘wid’ starting from ‘x’ & ‘y’ co-ordinate from the angle ‘start’ to angle ‘end’.
g.fillArc(100,100,250,250,0,360); – This method is used to fill the face circle with black color.
g.fillArc(170,185,25,25,0,360); – This method is used to fill the left eye circle with black color.
g.fillArc(255,185,25,25,0,360); – This method is used to fill the right eye circle with black color.
 - c) **setColor(Color.color name):** This method changes the color of the drawing for next called methods.
 Colors used in this program are Yellow and Black by using **setColor(Color.yellow);** and **setColor(Color.black);** methods.

advertisement

Runtime Test Cases

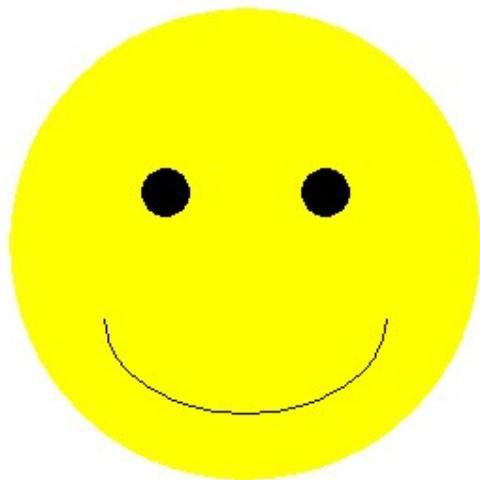
Here's the run time test case to draw a Smiling Face using methods of Graphics class.

Test case 1 – Here's the runtime output of the Smiling Face. Smiling Face should display a face with background color as yellow. The eyes are to be properly positioned and filled with color black. The smile is to be drawn with

black color with proper positioning.

 Smiley

— □ ×



132. Java Program to Display Several Dots on the Screen Continuously

[« Prev](#)

[Next »](#)

This is a Java program to display several dots on the frame continuously.

Problem Description

We have to write a program in Java such that it creates a frame and displays dots at random positions continuously using **paint** methods of **Graphics** class. Dots on frame continuously increases until the user closes the frame.

Expected Input and Output

For drawing dots on the frame at random positions, we can have the following sets of input and output.

advertisement

1. **To test the drawing of dots at random positions:** We expect a frame having a black background with the green dots at random positions.
2. **To test if number of dots are increasing:** It is expected that after some time, the number of dots on the frame continuously increases until the user closes the frame.

Problem Solution

1. Create a frame.
2. To draw dots continuously, repeatedly call function **paint**, until the frame has been closed, that is till the frame is visible.
3. Delay the function call of method **paint** using **Thread.sleep()** with duration as needed.
4. In method **paint**, generate two random points that lie within the limits of the frame using **Math.random()** method of **java.lang.Math** package.
5. Draw a point at the random point using method **drawLine()** of **Graphics** class.

advertisement

Program/Source Code

Here is source code of the Java Program to draw dots at random positions continuously using Graphics class. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.lang.Math;
4. import java.awt.Graphics.*;
5. class Infinite_Dots
6. {
7.     //Driver function
8.     public static void main(String args[])
9.     {
10.         //Create a frame
11.         JFrame frame=new JFrame("Infinite Dots");
12.         frame.setSize(500,500);
13.         frame.getContentPane().setBackground(Color.black);
14.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.         frame.setVisible(true);
16.         /*Draw the dots on frame continuously using method paint
17.          until the user closes the frame*/
18.         while(frame.isVisible())
19.         {
20.             paint(frame.getGraphics());
21.             try
22.             {
```

```

23.                                     //Delay by 1ms
24.                                     Thread.sleep(1);
25.                                 }
26.                             catch(InterruptedException ie){}
27.                         }
28. }
29. //function to draw a dot on the frame
30. public static void paint(Graphics g)
31. {
32.     g.setColor(Color.green);
33.     int x=(int)(Math.random()*1000)%500;
34.     int y=(int)(Math.random()*1000)%500;
35.     g.drawLine(x,y,x,y);
36. }
37.

```

Program Explanation

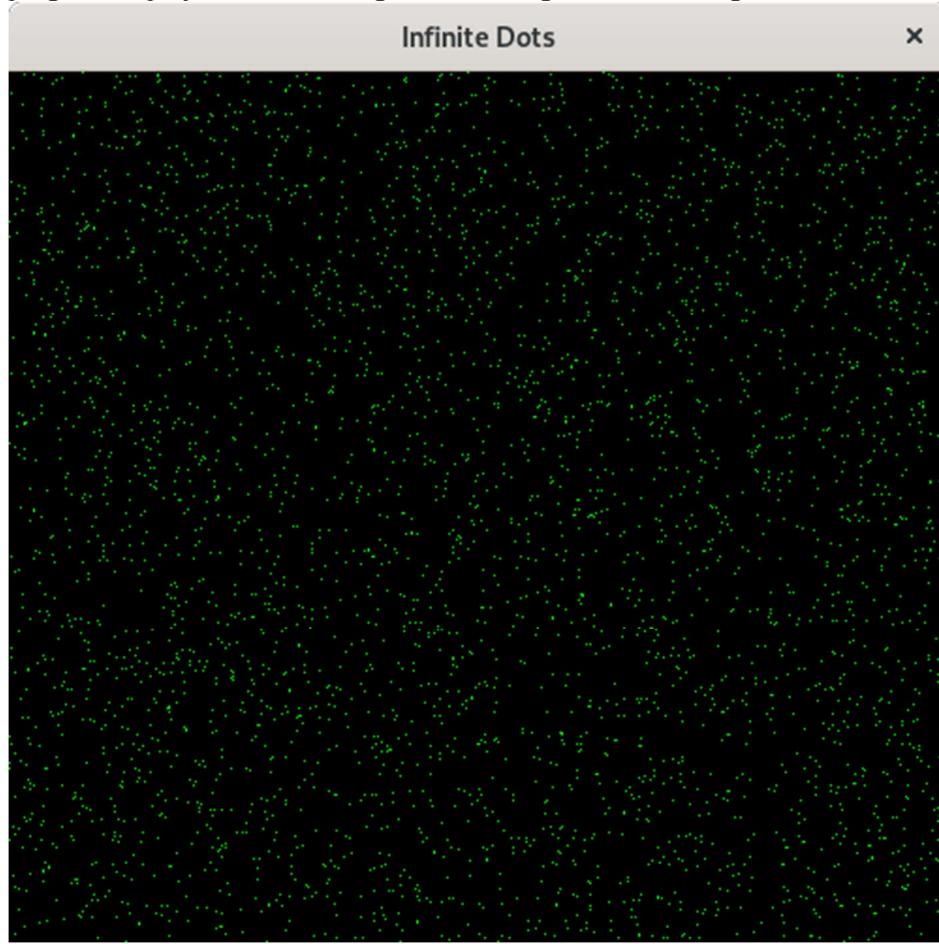
1. A frame is created with background color as black.
2. The method **paint** is repeatedly called to draw a dot until the frame is closed.
3. We delay the call of method **paint** by 1ms using **Thread.sleep()** to visualize the addition of new dots on the frame.
4. In method **paint** we change the color of graphics to green by using method **setColor(Color.color name)** defined in the **Graphics** class.
5. Generate a random co-ordinate (x,y) such that it lies within the bounds of the frame.
6. Method **drawLine(x1,y1,x2,y2)** draws a line from point (x1,y1) to point (x2,y2). To draw a dot we replace the points (x1,y1) & (x2,y2) by (x,y), that is the start and end positions of line is same as the randomly generated (x,y) co-ordinate.

advertisement

Runtime Test Cases

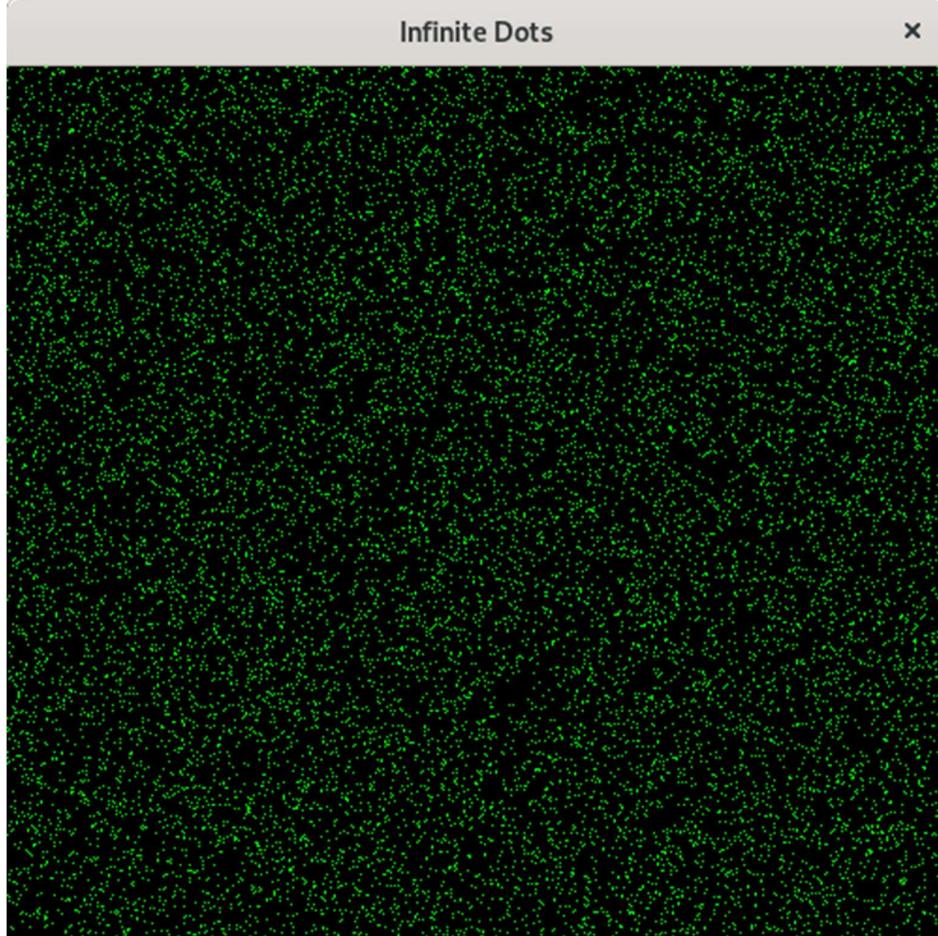
Here's the run time test cases to draw dots at several positions continuously.

Test case 1 – Here is the runtime output of the Java program to test the drawing dots at random positions. The program displays a frame having a black background with the green dots at random positions.



Test case 2 – Here is the runtime output of the Java program to test if number of dots have increased. The program displays a frame having a black background with the green dots at random positions. After some time,

the number of dots on the frame continuously increases until the user closes the frame.



133. Java Program to Display Text in the Frame by using `drawString` and Inheriting `JPanel` Class

« [Prev](#)

[Next](#) »

This is a Java program to display text in the frame by using `drawString` and inheriting `JPanel` class to override the `paintComponent` method.

Problem Description

We have to write a program in Java such that it creates a frame and draws a text using the `drawString` method of `Graphics` class by overriding the `paintComponent` method of the `JPanel` class.

Expected Input and Output

For drawing a text, we can have the following set of input and output.

advertisement

To test `drawString`: We expect a string “Hello World” on the frame. The string should have the **TimesRoman** font, with **Bold** letters and legible font size.

Problem Solution

1. Create a frame with white background.
2. Create an object of the class and add it to the frame. To add the object to the frame, inheriting the `JPanel` class is needed.
3. Display the frame.
4. Override the `paintComponent` method of the `JPanel` class using `@Override` keyword.
5. Set the color of the graphics object to black using `setColor(Color.color name)` method.
6. Create a new font as required by creating an object of class `Font`.
7. Change the font of the graphics object with the new font created.
8. Use method `drawString` to draw the string.

Program/Source Code

Here is source code of the Java Program to draw sting using methods of Graphics class and inheriting `JPanel` class. The program is successfully compiled and tested using javac compiler on Fedora 30.

advertisement

```
1. /* Java Program to draw a string using methods of Graphics class
2. and inheriting JPanel class to override the paintComponent */
3. import javax.swing.*;
4. import java.awt.*;
5. class Draw_Text extends JPanel
6. {
7.     //Driver function
8.     public static void main(String args[])
9.     {
10.         //Create a Frame
11.         JFrame frame=new JFrame("Draw a Text");
12.         frame.setSize(500,500);
13.         frame.getContentPane().setBackground(Color.white);
14.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.         //Create an object of the class
16.         Draw_Text obj=new Draw_Text();
17.         //Add object of class to frame
18.         frame.add(obj);
19.         //Display the frame
20.         frame.setVisible(true);
21.     }
22.     //Function to draw a text using drawString method
```

```

23. @Override
24. public void paintComponent(Graphics g)
25. {
26.     g.setColor(Color.black);
27.     Font myFont=new Font("TimesRoman",Font.BOLD,30);
28.     g.setFont(myFont);
29.     g.drawString("Hello World",100,225);
30. }
31.

```

Program Explanation

1. Create a class **Draw_Text** which inherits the **JPanel** class.
2. Create a frame with background color as white by using **setBackground(Color.white);**. **frame.setSize(500,500);** is used to set the width and height of the frame.
3. Create an object of the class **Draw_Text obj=new Draw_Text();** and add it to the frame by using **frame.add(obj);**.
4. **frame.setVisible(true);** is used to display the frame.
5. Override the **paintComponent** of the **JPanel** class using the **@Override** keyword.
6. Create a font **myFont** as an object of the class **Font** in the following way:
Font myFont = new Font("Font Name",Font.type,Font Size);
type means the kind of font required like **BOLD**, **ITALIC**, etc.
7. **myFont** values used in the program are:
myFont=new Font("TimesRoman",Font.BOLD,30);
“TimesRoman” is the Font Name, “Font.BOLD” is the Font.type and “30” is the Font Size of the string.
8. Set the created font with the graphics object using **setFont** method.
9. Use **drawString(str,len,wid)** method to draw the string “str” with the maximum length “len” and width “wid”. The values of drawString method is **drawString("Hello World",100,225);**. Here “Hello World” is the string, “100” is the maximum length of the string and “225” is the maximum width of the string.

Runtime Test Cases

Here's the run time test cases for drawing text using method of Graphics class and inheriting **paintComponent** of **JPanel** class.

Test case 1 – Here's the runtime output of the **drawString** method. The program displays the string “Hello World” on the frame with TimesRoman font, Bold letters and legible font size.



134. Java Program to know Which Fonts are Available in a Local System

[« Prev](#)

[Next »](#)

This is a Java program to know which fonts are available in a local system.

Problem Description

We have to write a program in Java such that it generates a list of fonts available in a Local System using **GraphicsEnvironment**.

Expected Input and Output

For knowing which fonts are available in a Local System, we can have the following set of input and output.

To know the fonts available: On the execution of the program, it is expected that a list is generated which contains the name of fonts available in the local system.

advertisement

Problem Solution

1. Create a variable to get the local graphics environment from **GraphicsEnvironment** class.

The **GraphicsEnvironment** class contains a collection of GraphicsDevice objects and Font objects available in a Local or Remote platform.

2. Get List of Font Family available using **getAvailableFontFamilyNames()** method.

3. Display the list.

Program/Source Code

Here is source code of the Java Program to get the list of font available in a local system. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. /* Java Program to get list of fonts available*/
2. import java.awt.GraphicsEnvironment;
3. class Fonts_Available
4. {
5.     //Driver Function
6.     public static void main(String[] args)
7.     {
8.         //Get the local graphics environment
9.         GraphicsEnvironment local_env;
10.        local_env= GraphicsEnvironment.getLocalGraphicsEnvironment();
11.        //Get available font names
12.        String allfonts[] = local_env.getAvailableFontFamilyNames();
13.        //Display the list of fonts
14.        for(int i=0;i<allfonts.length;i++)
15.            System.out.println(allfonts[i]);
16.    }
17. }
```

Program Explanation

1. Variable **local_env** gets the local graphics environment.

2. Use **getAvailableFontFamilyNames()** to get the fonts available and store it in an string array.

3. Display the array.

advertisement

Runtime Test Cases

Here's the run time test case for generating list of fonts available in a Local System.

Test case 1 – Here's the runtime output to get the list of fonts available. The program displays the list which contains the name of fonts available in the local system.

advertisement

```
$ java Fonts_Available
Abyssinica SIL
Bitstream Charter
C059
Caladea
Cantarell
Cantarell Extra Bold
Cantarell Light
Cantarell Thin
Carlito
cmex10
cmmi10
cmr10
cmsy10
Comfortaa
Comfortaa Light
Courier 10 Pitch
Cursor
D050000L
DejaVu Sans
DejaVu Sans Condensed
DejaVu Sans Light
DejaVu Sans Mono
DejaVu Serif
DejaVu Serif Condensed
Dialog
DialogInput
Droid Sans
Droid Sans Arabic
Droid Sans Armenian
Droid Sans Devanagari
Droid Sans Ethiopic
Droid Sans Fallback
Droid Sans Georgian
Droid Sans Hebrew
Droid Sans Japanese
Droid Sans Tamil
Droid Sans Thai
esint10
eufm10
FreeMono
FreeSans
FreeSerif
Jomolhari
Khmer OS
Khmer OS Content
Khmer OS System
Latin Modern Math
Liberation Mono
Liberation Sans
Liberation Serif
LM Mono 10
LM Mono 12
LM Mono 8
LM Mono 9
LM Mono Caps 10
LM Mono Light 10
LM Mono Light Cond 10
LM Mono Prop 10
LM Mono Prop Light 10
LM Mono Slanted 10
```

LM Roman 10
LM Roman 12
LM Roman 17
LM Roman 5
LM Roman 6
LM Roman 7
LM Roman 8
LM Roman 9
LM Roman Caps 10
LM Roman Demi 10
LM Roman Dunhill 10
LM Roman Slanted 10
LM Roman Slanted 12
LM Roman Slanted 17
LM Roman Slanted 8
LM Roman Slanted 9
LM Roman Unslanted 10
LM Sans 10
LM Sans 12
LM Sans 17
LM Sans 8
LM Sans 9
LM Sans Demi Cond 10
LM Sans Quot 8
Lohit Assamese
Lohit Bengali
Lohit Devanagari
Lohit Gujarati
Lohit Gurmukhi
Lohit Kannada
Lohit Odia
Lohit Tamil
Lohit Telugu
Meera
Mingzat
MnSymbol
Monospaced
Montserrat
Montserrat Black
Montserrat ExtraBold
Montserrat ExtraLight
Montserrat Light
Montserrat Medium
Montserrat SemiBold
Montserrat Thin
msam10
msbm10
Nimbus Mono PS
Nimbus Roman
Nimbus Sans
Nimbus Sans Narrow
Noto Color Emoji
Noto Sans CJK JP Black
Noto Sans CJK JP Bold
Noto Sans CJK JP DemiLight
Noto Sans CJK JP Light
Noto Sans CJK JP Medium
Noto Sans CJK JP Regular
Noto Sans CJK JP Thin
Noto Sans CJK KR Black
Noto Sans CJK KR Bold
Noto Sans CJK KR DemiLight
Noto Sans CJK KR Light
Noto Sans CJK KR Medium
Noto Sans CJK KR Regular
Noto Sans CJK KR Thin

Noto Sans CJK SC Black
Noto Sans CJK SC Bold
Noto Sans CJK SC DemiLight
Noto Sans CJK SC Light
Noto Sans CJK SC Medium
Noto Sans CJK SC Regular
Noto Sans CJK SC Thin
Noto Sans CJK TC Black
Noto Sans CJK TC Bold
Noto Sans CJK TC DemiLight
Noto Sans CJK TC Light
Noto Sans CJK TC Medium
Noto Sans CJK TC Regular
Noto Sans CJK TC Thin
Noto Sans Mono CJK JP Bold
Noto Sans Mono CJK JP Regular
Noto Sans Mono CJK KR Bold
Noto Sans Mono CJK KR Regular
Noto Sans Mono CJK SC Bold
Noto Sans Mono CJK SC Regular
Noto Sans Mono CJK TC Bold
Noto Sans Mono CJK TC Regular
Noto Sans Sinhala
Noto Sans Sinhala Blk
Noto Sans Sinhala Cond
Noto Sans Sinhala Cond Blk
Noto Sans Sinhala Cond ExtBd
Noto Sans Sinhala Cond ExtLt
Noto Sans Sinhala Cond Light
Noto Sans Sinhala Cond Med
Noto Sans Sinhala Cond SemBd
Noto Sans Sinhala Cond Thin
Noto Sans Sinhala ExtBd
Noto Sans Sinhala ExtCond
Noto Sans Sinhala ExtCond Blk
Noto Sans Sinhala ExtCond ExtBd
Noto Sans Sinhala ExtCond ExtLt
Noto Sans Sinhala ExtCond Light
Noto Sans Sinhala ExtCond Med
Noto Sans Sinhala ExtCond SemBd
Noto Sans Sinhala ExtCond Thin
Noto Sans Sinhala ExtLt
Noto Sans Sinhala Light
Noto Sans Sinhala Med
Noto Sans Sinhala SemBd
Noto Sans Sinhala SemCond
Noto Sans Sinhala SemCond Blk
Noto Sans Sinhala SemCond ExtBd
Noto Sans Sinhala SemCond ExtLt
Noto Sans Sinhala SemCond Light
Noto Sans Sinhala SemCond Med
Noto Sans Sinhala SemCond SemBd
Noto Sans Sinhala SemCond Thin
Noto Sans Sinhala Thin
Noto Serif CJK JP
Noto Serif CJK JP Black
Noto Serif CJK JP ExtraLight
Noto Serif CJK JP Light
Noto Serif CJK JP Medium
Noto Serif CJK JP SemiBold
Noto Serif CJK KR
Noto Serif CJK KR Black
Noto Serif CJK KR ExtraLight
Noto Serif CJK KR Light
Noto Serif CJK KR Medium
Noto Serif CJK KR SemiBold

Noto Serif CJK SC
Noto Serif CJK SC Black
Noto Serif CJK SC ExtraLight
Noto Serif CJK SC Light
Noto Serif CJK SC Medium
Noto Serif CJK SC SemiBold
Noto Serif CJK TC
Noto Serif CJK TC Black
Noto Serif CJK TC ExtraLight
Noto Serif CJK TC Light
Noto Serif CJK TC Medium
Noto Serif CJK TC SemiBold
Nuosu SIL
OpenSymbol
P052
Padauk
PakType Naskh Basic
PT Sans
PT Sans Narrow
rsfs10
SansSerif
Serif
Source Code Pro
Source Code Pro Black
Source Code Pro ExtraLight
Source Code Pro Light
Source Code Pro Medium
Source Code Pro Semibold
STIX
stmary10
Symbola
TeX Gyre Adventor
TeX Gyre Bonum
TeX Gyre Bonum Math
TeX Gyre DejaVu Math
TeX Gyre Pagella
TeX Gyre Pagella Math
TeX Gyre Schola Math
TeX Gyre Termes Math
TeXGyreChorus
TeXGyreCursor
TeXGyreHeros
TeXGyreHerosCn
TeXGyreSchola
TeXGyreTermes
URW Bookman
URW Gothic
Utopia
Waree
wasy10
Z003

135. Java Program to Get and Set State and Get Label of a CheckBox

[« Prev](#)

[Next »](#)

This is a Java program to get and set state and get label of a checkbox.

Problem Description

We have to write a program in Java such that it creates a checkbox to get its present state, change/set its state and get its label.

Expected Input and Output

For Get and Set State and Get Label of a CheckBox, we can have the following 5 different sets of input and output.

advertisement

1. To get the un-checked state:

For example:

If the checkbox is in un-checked state,
the expected output is "Status of Checkbox : Un-Checked"

2. To get the checked state:

For example:

If the checkbox is in checked state,
the expected output is "Status of Checkbox : Checked"

3. To get the set state from un-checked to checked:

For example:

advertisement

If the state of checkbox is changed from un-checked to checked,
the expected output is "Status of Checkbox Changed from Un-Checked to Checked"

4. To get the set state from checked to un-checked:

For example:

If the state of checkbox is changed from checked to un-checked,
the expected output is "Status of Checkbox Changed from Checked to Un-Checked"

5. To get the label of checkbox:

For example:

advertisement

If the get label option is selected,
the expected output is "Label of Checkbox : Sample Checkbox"

Problem Solution

1. Create a frame, a text field, 3 buttons – Get State, Set State and Get Label and a Checkbox.
2. Associate **ActionListener** with the 3 buttons.
 - a) Method **get_State** is called when the **Get State** button is clicked.
 - b) Method **set_State** is called when the **Set State** button is clicked.
 - c) Method **get_Label** is called when the **Get Label** button is clicked.
3. Use function **getState()** to get the present state of the checkbox.
4. Use function **setState()** to set the state of the checkbox.
5. Use function **getLabel()** to get the label of the checkbox.
6. Display appropriate message in the text fields.

Program/Source Code

Here is source code of the Java Program to Get and Set State and Get Label of a CheckBox. The program is successfully compiled and tested using javac compiler on Fedora 30.

advertisement

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4. class Check_Box implements ActionListener
5. {
6.     static JTextField text;
7.     static Checkbox checkbox;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame=new JFrame("Check Box");
13.        frame.setSize(500,500);
14.        frame.setBackground(Color.white);
15.        frame.setLayout(null);
16.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17.        //Create a textfield
18.        text=new JTextField();
19.        text.setBounds(0,0,500,50);
20.        frame.add(text);
21.        //Create 3 buttons
22.        JButton get_st=new JButton("Get State");
23.        JButton set_st=new JButton("Set State");
24.        JButton get_lb=new JButton("Get Label");
25.        get_st.setBounds(50,80,100,50);
26.        set_st.setBounds(180,80,100,50);
27.        get_lb.setBounds(310,80,100,50);
28.        frame.add(get_st);
29.        frame.add(set_st);
30.        frame.add(get_lb);
31.        //Create an object of class
32.        Check_Box obj=new Check_Box();
33.        //Associate ActionListener with the buttons
34.        get_st.addActionListener(obj);
35.        set_st.addActionListener(obj);
36.        get_lb.addActionListener(obj);
37.        //Create a checkbox
38.        checkbox=new Checkbox("Sample Checkbox");
39.        checkbox.setBounds(150,200,200,80);
40.        frame.add(checkbox);
41.        //Display the frame
42.        frame.setVisible(true);
43.    }
44.    //function to call different methods based on user's choice of button
45.    @Override
46.    public void actionPerformed(ActionEvent e)
47.    {
48.        String option=e.getActionCommand();
49.        if(option.equals("Get State"))
50.            get_State();
51.        else if(option.equals("Set State"))
52.            set_State();
53.        else
54.            get_Label();
55.    }
56.    //function to get the current state of the checkbox
57.    public void get_State()
58.    {
59.        boolean state=checkbox.getState();
60.        if(state==true)
61.            text.setText("State of Checkbox : Checked");
```

```

62.     else
63.         text.setText("State of Checkbox : Un-Checked");
64.     }
65. //function to change the current state of the checkbox
66. public void set_State()
67. {
68.     text.setText("State of Checkbox changed from ");
69.     boolean state=checkbox.getState();
70.     if(state==true)
71.     {
72.         checkbox.setState(false);
73.         text.setText(text.getText()+"Checked to Un-Checked");
74.     }
75.     else
76.     {
77.         checkbox.setState(true);
78.         text.setText(text.getText()+"Un-Checked to Checked");
79.     }
80. }
81. //function to get the label of the checkbox
82. public void get_Label()
83. {
84.     text.setText("Label of the checkbox is : ");
85.     text.setText(text.getText()+checkbox.getLabel());
86. }
87.

```

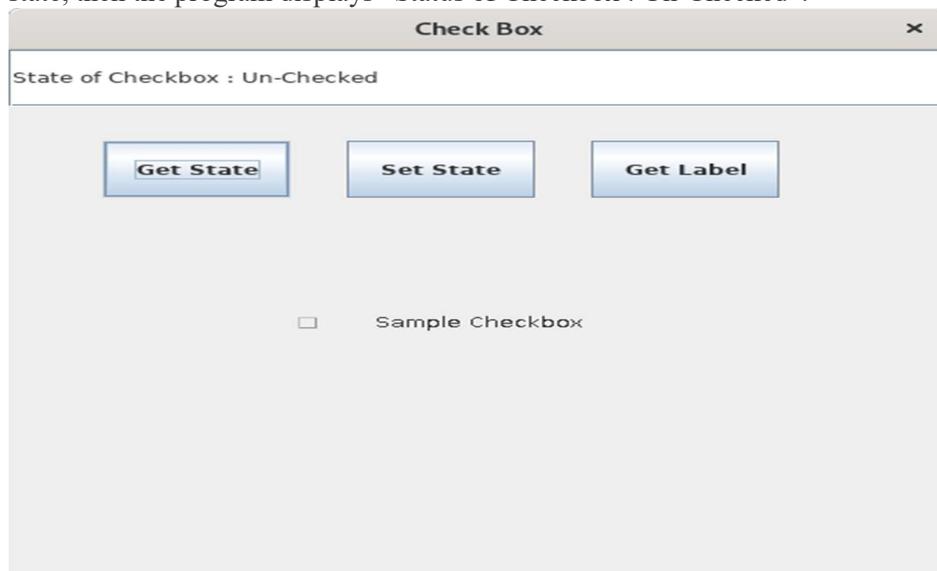
Program Explanation

1. The **actionPerformed(ActionEvent)** is called when any of the buttons is clicked. Use function **getActionCommand()** to know which button was clicked and call the respective methods.
2. In method **get_State()** display the present state of the checkbox.
3. In method **set_State()** change the current state of checkbox and display the necessary message.
4. In method **get_Label()** display the label of the checkbox.

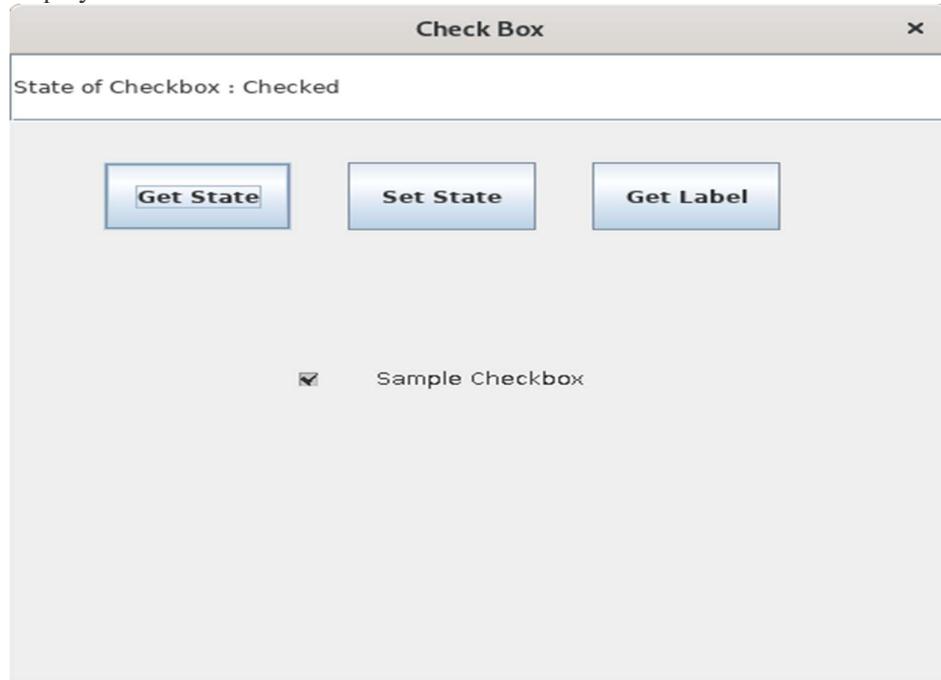
Runtime Test Cases

Here's the run time test cases for Get and Set State and Get Label of a CheckBox for 5 different input cases.

Test case 1 – Here's the runtime output to get the un-checked state. For example, the checkbox is in un-checked state, then the program displays “Status of Checkbox : Un-Checked”.

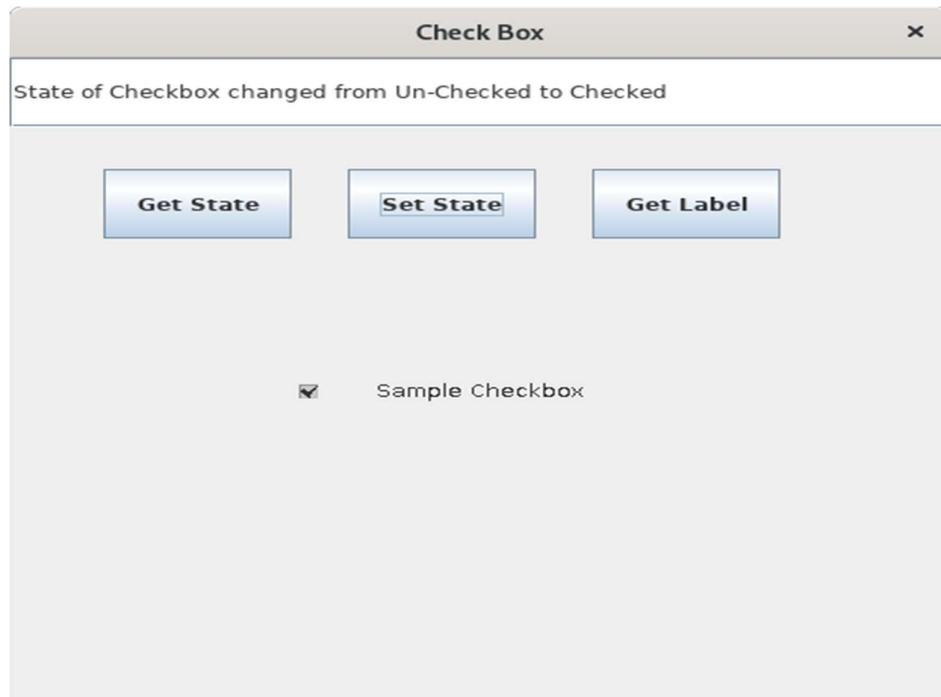


Test case 2 – Here's the runtime output to get the checked state. If user selects the checkbox, then the program displays “Status of Checkbox : Checked”.



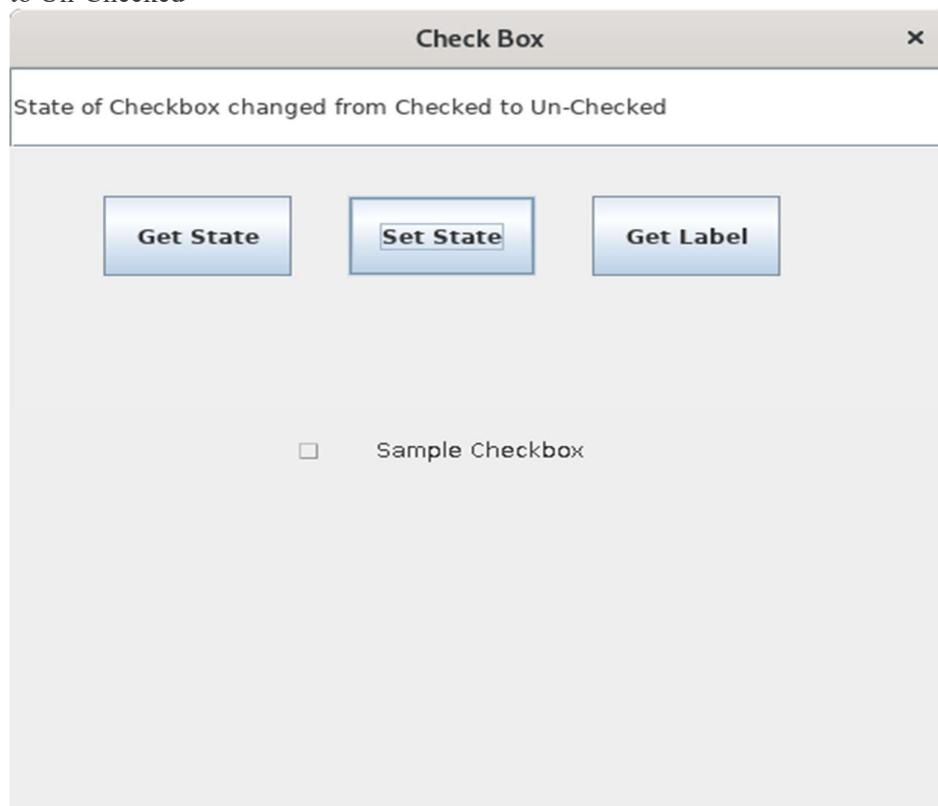
advertisement

Test case 3 – Here's the runtime output to set the state from un-checked to checked. If we change the state of checkbox from un-checked to checked, then the program displays “Status of Checkbox Changed from Un-Checked to Checked”.

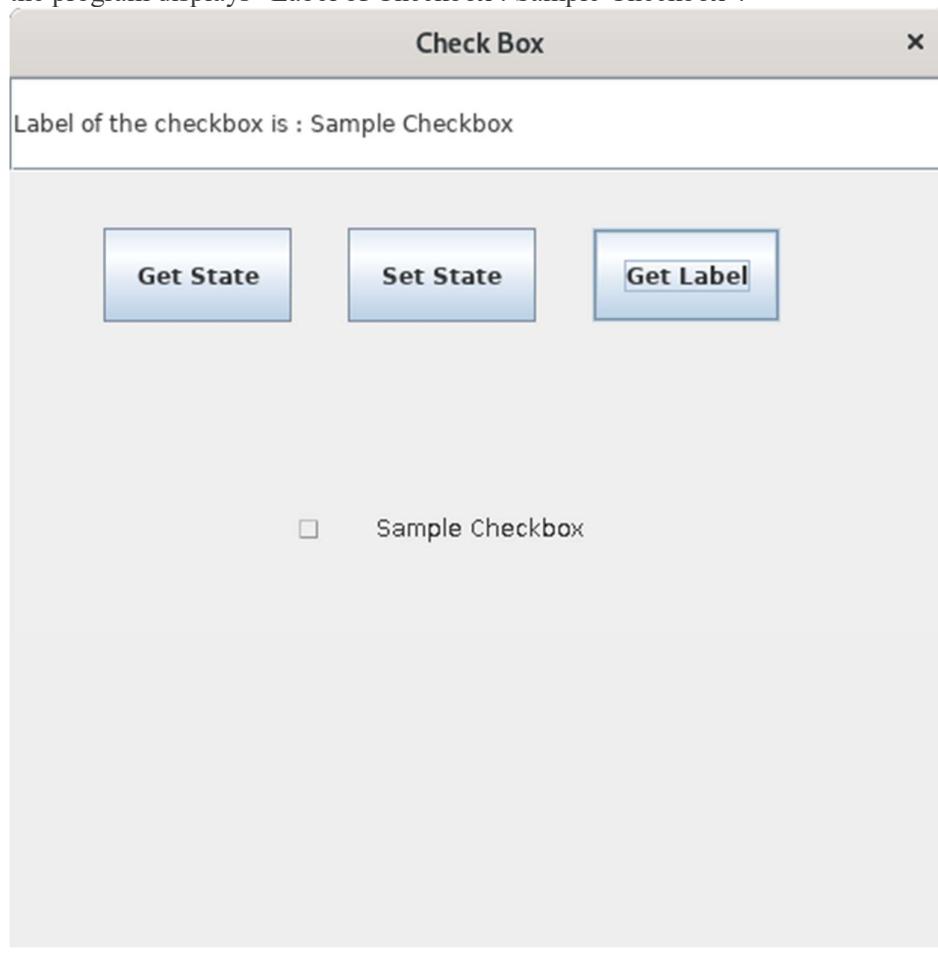


Test case 4 – Here's the runtime output to set the state from checked to un-checked. If we change the state of checkbox from checked to un-checked, then the program displays “Status of Checkbox Changed from Checked

to Un-Checked”



Test case 5 – Here’s the runtime output to get the label of the checkbox. If user selects the get label option, then the program displays “Label of Checkbox : Sample Checkbox”.



136. Java Program to Create 2 Radio Buttons ‘Yes’ and ‘No’ and Display the Selected Button Label

[« Prev](#)

[Next »](#)

This is a Java program to create 2 radio buttons ‘yes’ and ‘no’ and display the selected button label.

Problem Description

We have to write a program in Java such that it creates 2 Radio Buttons – ‘Yes’ and ‘No’ and displays the label of the button selected in a TextField.

Expected Input and Output

For getting the label of 2 radio buttons, we can have the following sets of input and output.

advertisement

1. To get the label of ‘Yes’ button:

When the radio button 'Yes' is selected,
then the expected output is "Label of Button Selected : Yes"

2. To get the label of ‘No’ button:

advertisement

When the radio button 'No' is selected,
then the expected output is "Label of Button Selected : No"

Problem Solution

1. Create a frame, a text field, and two radio buttons (Yes & No).
2. Associate **ActionListener** with the two radio buttons.
3. In method **actionPerformed** when a radio button is selected check the status of the other radio button and change it to un-selected as two radio buttons cannot be selected at any time.
4. Display the message accordingly.

Program/Source Code

Here is source code of the Java Program to create two radio buttons and get the label of button selected. The program is successfully compiled and tested using javac compiler on Fedora 30.

advertisement

```
1. /*Java Program to Create Two Radio Buttons and Get the Label of Button Selected*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Radio_Button implements ActionListener
6. {
7.     static JRadioButton yes;
8.     static JRadioButton no;
9.     static JTextField text;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.        //Create a frame
14.        JFrame frame=new JFrame("Radio Button");
15.        frame.setSize(500,500);
16.        frame.setLayout(null);
17.        frame.setBackground(Color.white);
18.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19.        //Create a text field
```

```

20.     text=new JTextField();
21.     text.setBounds(0,0,500,50);
22.     frame.add(text);
23.     //Create two radio buttons
24.     yes=new JRadioButton("Yes");
25.     no=new JRadioButton("No");
26.     yes.setBounds(210,100,80,60);
27.     no.setBounds(210,200,80,60);
28.     frame.add(yes);
29.     frame.add(no);
30.     //Create an object of the class
31.     Radio_Button obj=new Radio_Button();
32.     //Associate ActionListener with the radio buttons
33.     yes.addActionListener(obj);
34.     no.addActionListener(obj);
35.     //Display the frame
36.     frame.setVisible(true);
37. }
38. //function to display the label of button selected
39. public void actionPerformed(ActionEvent e)
40. {
41.     String b=e.getActionCommand();
42.     if(b.equals("Yes"))
43.     {
44.         if(no.isSelected())
45.             no.setSelected(false);
46.     }
47.     else
48.     {
49.         if(yes.isSelected())
50.             yes.setSelected(false);
51.     }
52.     text.setText("Label of Button Selected : "+b);
53. }
54. }
```

Program Explanation

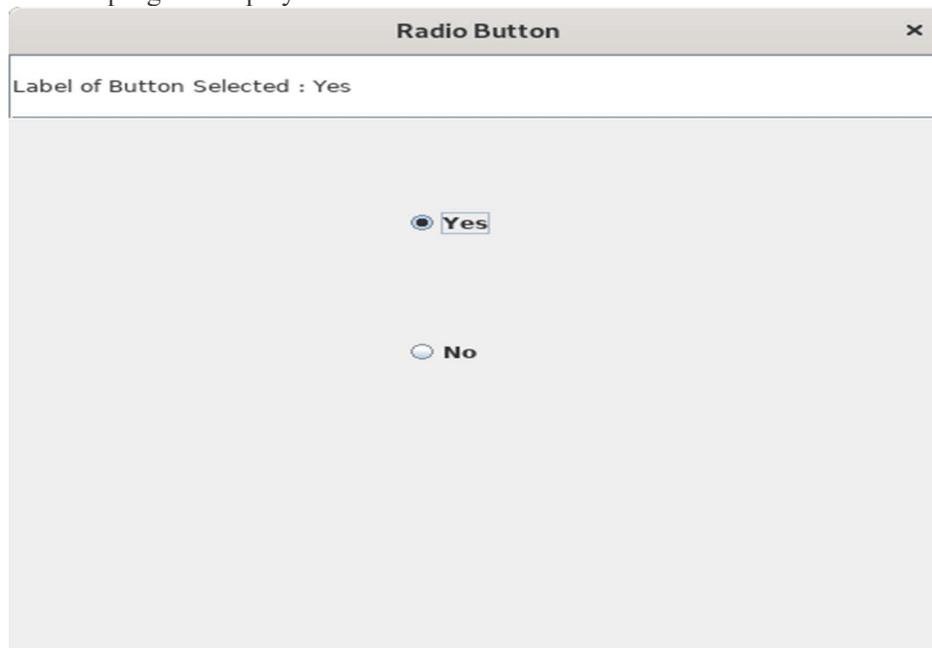
1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.
2. Create an text field with **setBounds(int x, int y, int width, int height)**, where ‘x’ & ‘y’ is used to set the position of top-left corner. ‘int width’ & ‘int height’ is used to set the width and height of the exit button.
3. Here the text field values of set bounds are **text.setBounds(0,0,500,50);**, where ‘0’ & ‘0’ is used to set the position of button from top and left side of the frame. ‘500’ & ‘50’ is used to set the width and height of the exit button.
4. Create two radio buttons Yes & No by setting setbound values as follows:
yes.setBounds(210,100,80,60); -> Values to create Yes Radio Button
no.setBounds(210,200,80,60); -> Values to create No Radio Button
5. **Radio_Button obj=new Radio_Button();** is used to create an object of the class.
6. Once object of the class is created. Associate ActionListener with the two radio buttons by using **yes.addActionListener(obj);** and **no.addActionListener(obj);**
7. **frame.setVisible(true);** is used to display the frame.
8. In method **actionPerformed** when a radio button is selected check the status of the other radio button and change it to un-selected as two radio buttons cannot be selected at any time. ActionEvent class is used to perform component defined action.
9. To check which button was clicked, and change the status of the other radio button to un-selected if it was selected using functions **isSelected** and **setSelected** respectively.
10. If any radio button is selected. Display appropriate message in the text box.

Runtime Test Cases

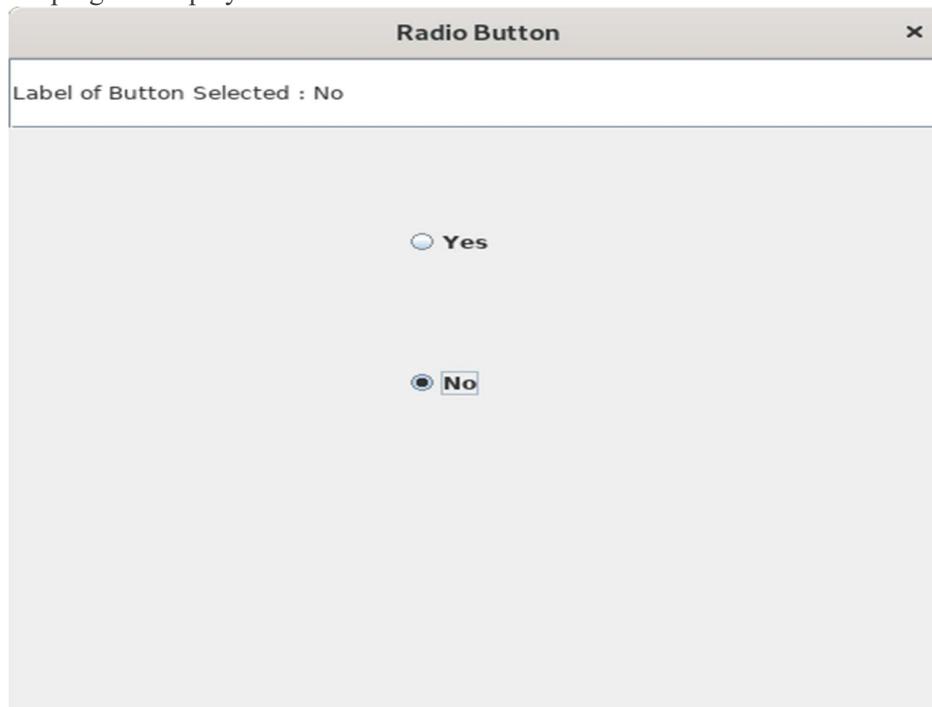
Here's the run time test cases for getting the label of the radio button selected.

advertisement

Test case 1 – Here's the runtime output to get the label of ‘Yes’ button. If user selects the radio button ‘Yes’, then the program displays “Label of Button Selected : Yes”.



Test case 2 – Here's the runtime output to get the label of ‘No’ button. If user selects the radio button ‘No’, then the program displays “Label of Button Selected : No”.



137. Java Program to Create a Choice Menu to Select an Item and Display it in the Frame

[« Prev](#)

[Next »](#)

This is a Java program to create a choice menu to select an item and display it in the frame.

Problem Description

We have to write a program in Java such that it creates a choice menu consisting of 5 languages (C, C++, Java, Python, R), and the selected language from the choice menu is displayed in the frame.

Expected Input and Output

For displaying the language selected from a Choice Menu, we can have the following 5 sets of input and output.

advertisement

1. Language selected ‘C’:

If the language C is selected from the Choice Menu (C, C++, Java, Python, R), then the expected output is "Language Selected : C"

2. Language selected ‘C++’:

If the language C++ is selected from the Choice Menu (C, C++, Java, Python, R), then the expected output is "Language Selected : C++"

3. Language selected ‘Java’:

advertisement

If the language Java is selected from the Choice Menu (C, C++, Java, Python, R), then the expected output is "Language Selected : Java"

4. Language selected ‘Python’:

If the language Python is selected from the Choice Menu (C, C++, Java, Python, R), then the expected output is "Language Selected : Python"

5. Language selected ‘R’:

advertisement

If the language R is selected from the Choice Menu (C, C++, Java, Python, R), then the expected output is "Language Selected : R"

Problem Solution

1. Create a frame, a choice menu consisting of the 5 languages, and an output text field.
2. Add the choice menu to the frame.
3. Associate **ItemListener** with the choice menu, and display the item selected.

Program/Source Code

Here is source code of the Java Program to create a choice menu and display the item selected. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. /*Java Program to create a choice menu and display the item selected*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Choice_Menu implements ItemListener
6. {
7.     static Choice menu;
8.     static JTextField text;
9.     //Driver function
10.    public static void main(String args[])
11. }
```

```

11. {
12.     //Create a frame
13.     JFrame frame=new JFrame("Choice Menu");
14.     frame.setSize(500,500);
15.     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.     frame.setLayout(null);
17.     frame.getContentPane().setBackground(Color.white);
18.     //Create a choice menu
19.     menu=new Choice();
20.     menu.setBounds(100,100,120,60);
21.     frame.add(menu);
22.     //Add items to the menu
23.     menu.add("C");
24.     menu.add("C++");
25.     menu.add("Java");
26.     menu.add("Python");
27.     menu.add("R");
28.     //Create a text field for output
29.     text=new JTextField();
30.     text.setBounds(0,400,500,100);
31.     frame.add(text);
32.     //Create an object
33.     Choice_Menu obj=new Choice_Menu();
34.     //Associate ItemListener with the choice menu
35.     menu.addItemListener(obj);
36.     //Display the frame
37.     frame.setVisible(true);
38. }
39. //function to display the item selected
40. public void itemStateChanged(ItemEvent e)
41. {
42.     String item=menu.getSelectedItem();
43.     text.setText("Language Selected : "+item);
44. }
45. }
```

Program Explanation

1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.

2. **frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);** is a default close operation for the frame.

3. Create a choice menu with a **setBounds()** values and add it to the frame by using **frame.add();**

menu=new Choice(); → Used to create the choice menu.

menu.setBounds(100,100,120,60); → Sets the position of menu.

frame.add(menu); → It will add the menu to the frame.

advertisement

4. Once the menu is created. Add the items to the menu by using **menu.add(" ");**. The menu items used in this program are:

menu.add("C"); → Item “C” added to the menu.

menu.add("C++"); → Item “C++” added to the menu.

menu.add("Java"); → Item “Java” added to the menu.

menu.add("Python"); → Item “Python” added to the menu.

menu.add("R"); → Item “R” added to the menu.

5. Create a text field with a **setBounds()** values and add it to the frame. This text field is used for output to display a selected item from the menu.

6. **Choice_Menu obj=new Choice_Menu();** is used to create the object. Once object is created associate **ItemListener** with the choice menu.

7. **frame.setVisible(true);** is used to display the frame.

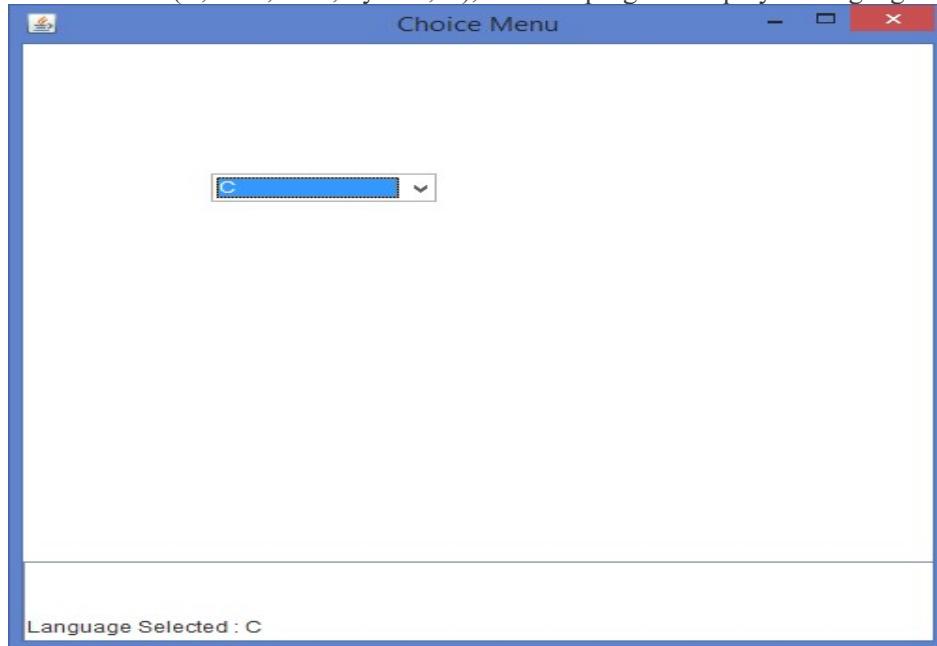
8. **String item=menu.getSelectedItem();** Function **getSelectedItem** returns the selected item in String format.

9. **text.setText("Language Selected : "+item);** → It displays the selected menu item from choice menu in the outputtextfield.

Runtime Test Cases

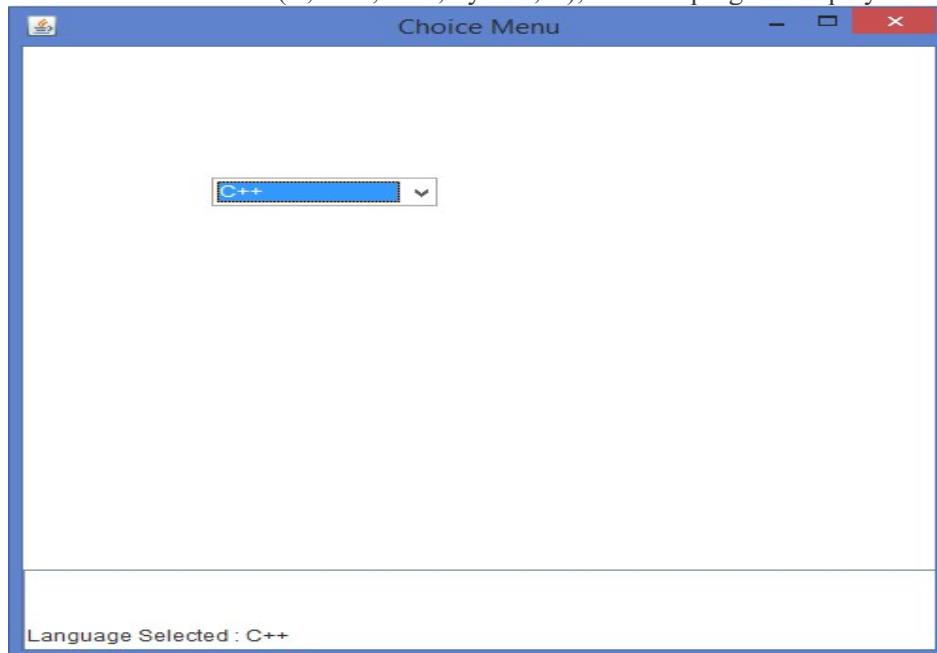
Here's the run time test case for getting the item selected from a choice menu.

Test case 1 – Here's the runtime output to test language ‘C’. For example, if the language C is selected from the Choice Menu (C, C++, Java, Python, R), then the program displays “Language Selected : C”.

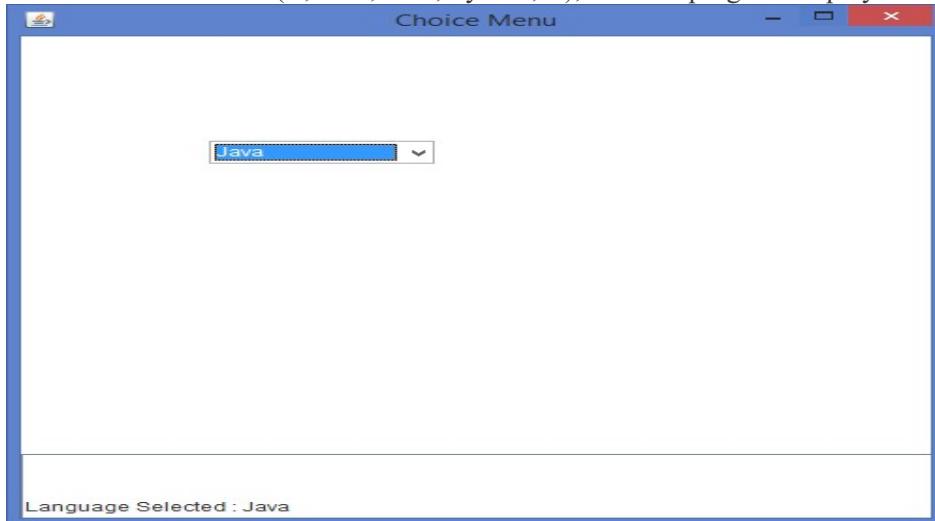


advertisement

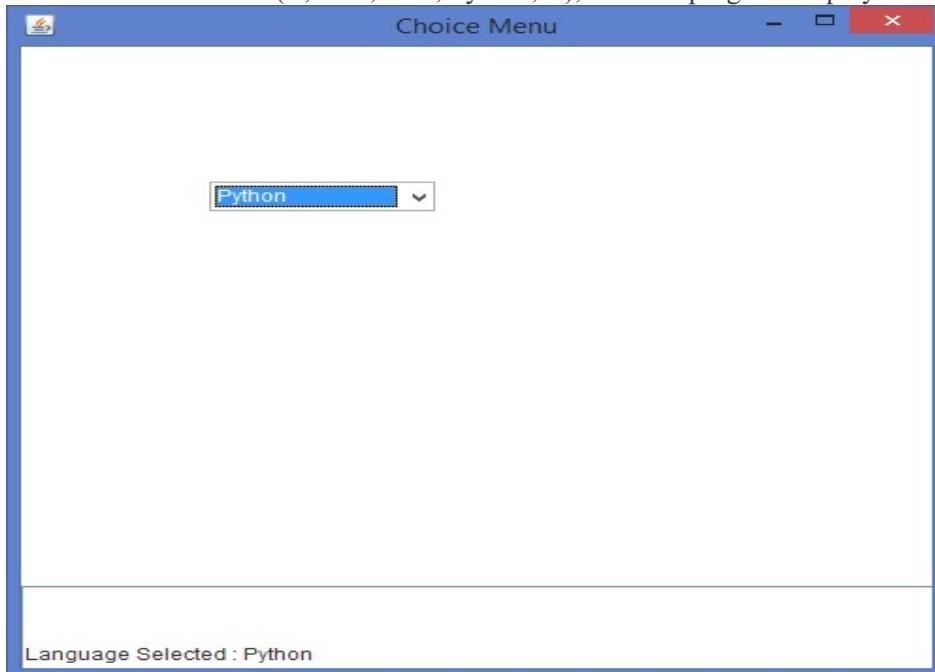
Test case 2 – Here's the runtime output to test language ‘C++’. For example, if the language C++ is selected from the Choice Menu (C, C++, Java, Python, R), then the program displays “Language Selected : C++”.



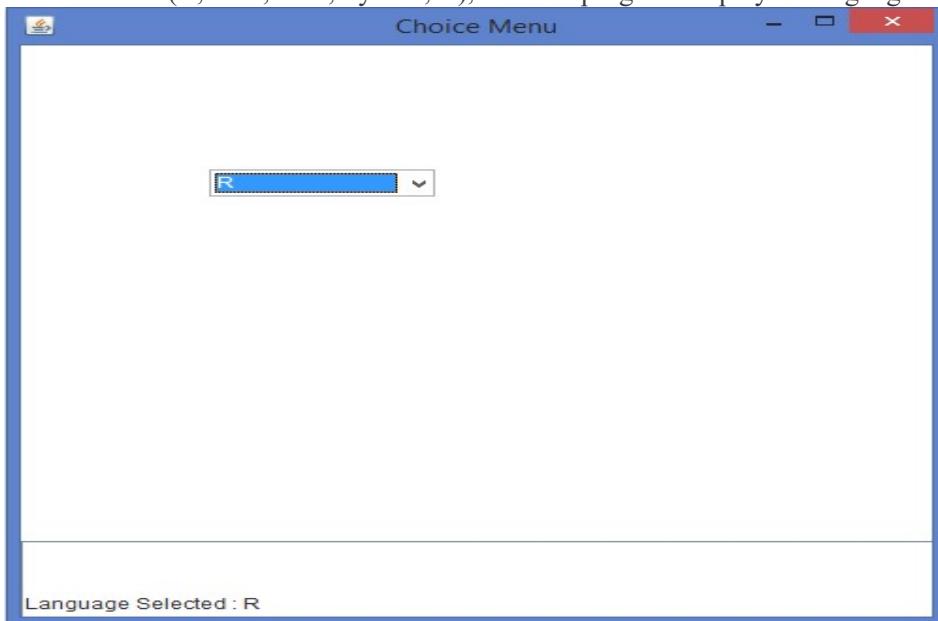
Test case 3 – Here’s the runtime output to test language ‘Java’. For example, if the language Java is selected from the Choice Menu (C, C++, Java, Python, R), then the program displays “Language Selected : Java”.



Test case 4 – Here’s the runtime output to test language ‘Python’. For example, if the language Python is selected from the Choice Menu (C, C++, Java, Python, R), then the program displays “Language Selected : Python”.



Test case 5 – Here's the runtime output to test language ‘R’. For example, if the language R is selected from the Choice Menu (C, C++, Java, Python, R), then the program displays “Language Selected : R”.



138. Java Program to Create a Vertical Scrollbar with Scroll Button Length 30px

[« Prev](#)

[Next »](#)

This is a Java program to create a vertical scrollbar with a scroll button length 30px.

Problem Description

We have to write a program in Java such that it creates a vertical scrollbar whose starting and ending positions are 0px and 400px with scroll button length 30px.

Expected Input and Output

For creating a vertical scrollbar, we can have the following set of input and output.

advertisement

To Create a Vertical Scrollbar :

At the execution of the program, it is expected that the frame consists of a vertical scrollbar with the given parameters (0px, 400px) with scroll button length 30px.

Problem Solution

1. Create a frame.
2. Create a scrollbar with vertical orientation and scroll size 30px.
3. Set the minimum and maximum value of scroll bar to 0 and 400 respectively.
4. Add the scrollbar to frame, and display the frame.

Program/Source Code

Here is source code of the Java Program to create a vertical scrollbar. The program is successfully compiled and tested using javac compiler on Fedora 30.

advertisement

```
1. /*Java Program to create a vertical scrollbar*/
2. import javax.swing.*;
3. import java.awt.*;
4. class Vertical_ScrollBar
5. {
6.     //Driver function
7.     public static void main(String args[])
8.     {
9.         //Create a frame
10.        JFrame frame=new JFrame("Vertical Scrollbar");
11.        frame.setSize(500,500);
12.        frame.setLayout(null);
13.        frame.getContentPane().setBackground(Color.white);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        //Create a Scrollbar
16.        Scrollbar scroll=new Scrollbar();
17.        scroll.setOrientation(Scrollbar.VERTICAL);
18.        scroll.setBounds(225,0,50,400);
19.        scroll.setMaximum(400);
20.        scroll.setMinimum(0);
21.        //Set size of scroll
22.        scroll.setVisibleAmount(30);
23.        //Add scrollbar to frame
24.        frame.add(scroll);
```

```
25.         //Display the frame  
26.         frame.setVisible(true);  
27.     }  
28. }
```

Program Explanation

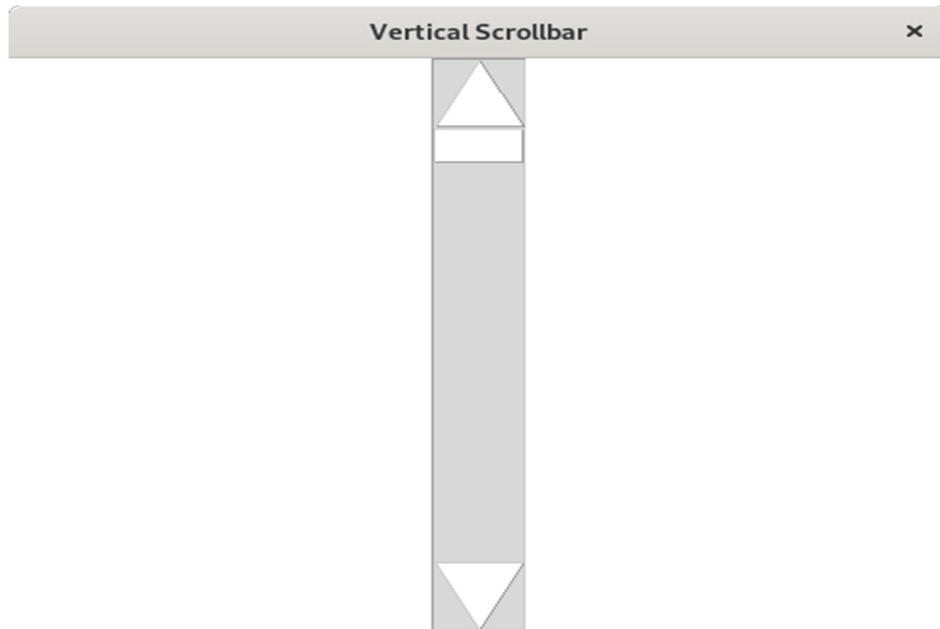
1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.
 2. **frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);** is a default close operation for the frame.
 3. Create a scrollbar and add it to the frame by using **frame.add();**
- Scrollbar scroll=new Scrollbar();** -> Used to create the scrollbar.
4. **scroll.setOrientation(Scrollbar.VERTICAL);** -> **setOrientation** is used to set the orientation of the scrollbar to vertical or horizontal. The orientation of the scrollbar used in this program is **vertical**.
 5. **scroll.setBounds(225,0,50,400);** -> Sets the position of scrollbar.
 6. **scroll.setMaximum(400);** -> **setMaximum** is used to set the maximum value of the scrollbar. The maximum value the scrollbar in this program is **400px**.
 7. **scroll.setMinimum(0);** -> **setMinimum** is used to set the minimum value of the scrollbar. The minimum value the scrollbar in this program is **0px**.
 8. **scroll.setVisibleAmount(30);** -> **setVisibleAmount** is used to set the size of the scroll. The size of the scroll in this program is **30px**.
 9. **frame.add(scroll);** -> It is used to add the scrollbar to the frame.
 10. **frame.setVisible(true);** is used to display the frame.

Runtime Test Cases

Here's the run time test case for creating a vertical scrollbar.

advertisement

Test case – Here's the runtime output to view the scrollbar. Once the program is executed, it displays the frame which consists of a vertical scrollbar with the given parameters (0px, 400px) with scroll button length 30px.



139. Java Program to Create a JTree with a Root Node and Other Nodes Spanning from Root Node

[« Prev](#)

[Next »](#)

This is a Java Program to Create a JTree with a Root Node and Other Nodes Spanning from Root Node

Problem Description

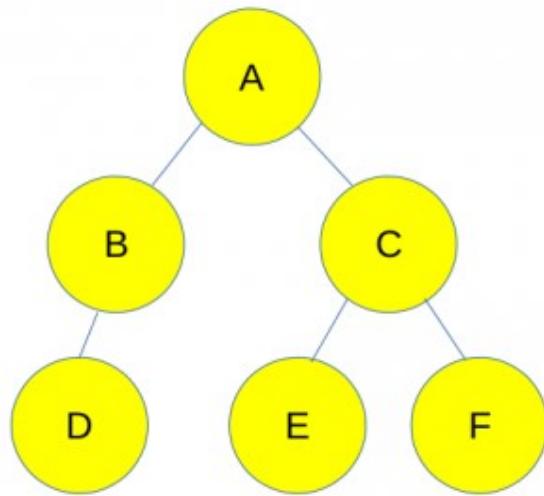
We have to write a program in Java such that it creates a tree with a root node and nodes spanning from it.

Expected Input and Output

For creating a JTree, we can have the following set of input and output.

advertisement

Consider, we want to create the tree given below :



To create a tree:

On the execution of the program,
it is expected that the tree is created and displayed on the frame.

Problem Solution

1. Create a root node ‘A’.
2. Create nodes ‘B’ and ‘C’, and add them to their parent node ‘A’.
3. Create node ‘D’, and add it to its parent node ‘B’.
4. Create nodes ‘E’ and ‘F’, and add them to their parent node ‘C’.
5. Create a tree with the root node ‘A’, and add the tree to frame.
6. Display the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to create a tree using JTree. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to create a tree*/
2. import javax.swing.*;
```

```

3. import java.awt.*;
4. import javax.swing.JTree;
5. import javax.swing.tree.*;
6. class Tree
7. {
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame = new JFrame("Tree");
13.        frame.setSize(500,500);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        //Create root 'A'
16.        DefaultMutableTreeNode A=new DefaultMutableTreeNode("A");
17.        //Create children 'B' & 'C'
18.        DefaultMutableTreeNode B=new DefaultMutableTreeNode("B");
19.        DefaultMutableTreeNode C=new DefaultMutableTreeNode("C");
20.        A.add(B);
21.        A.add(C);
22.        //Create child 'D' of B
23.        DefaultMutableTreeNode D=new DefaultMutableTreeNode("D");
24.        B.add(D);
25.        //Create children 'E' and 'F' of C
26.        DefaultMutableTreeNode E=new DefaultMutableTreeNode("E");
27.        DefaultMutableTreeNode F=new DefaultMutableTreeNode("F");
28.        C.add(E);
29.        C.add(F);
30.        //Create a tree
31.        JTree tree=new JTree(A);
32.        frame.add(tree);
33.        //Display the frame
34.        frame.setVisible(true);
35.    }
36.

```

Program Explanation

1. To create a node use the **DefaultMutableTreeNode** class.
 2. To create a tree use **JTree** class and specify the root node of the tree.
- advertisement

Runtime Test Cases

Here's the run time test case for creating a tree using JTree.

Test case – To view the tree.



140. Java Program to Change the Background Colour of Applet by Clicking on Control Button

[« Prev](#)

[Next »](#)

This is a Java Program to Change the Background Colour of Applet by Clicking on Control Button

Problem Description

We have to write a program in Java such that it changes the background color of the frame to a random color every time the control key on the keyboard is pressed.

Expected Input and Output

For changing the background color of frame, we have the following different sets of input and output.

advertisement

1. Initial Background Color of Applet :

When the program is executed,
it is expected that the applet initially has background color white.

2. To Change the Background Color of Applet :

advertisement

Everytime the Control key on the keyboard is pressed,
it is expected that the background color changes.

Problem Solution

1. Initially, set the background color as white.
2. Add **KeyListener** to the applet.
3. When the control key is pressed, change the background color of applet to a random color.

Program/Source Code

Here is source code of the Java Program to change background color of applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to Change Background Color of Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. import java.util.*;
6. public class Change_Background extends Applet implements KeyListener
7. {
8.     //Function to initialize the applet
9.     public void init()
10.    {
11.        setBackground(Color.white);
12.        addKeyListener(this);
13.    }
14.    //Function to change background color of applet
15.    public void keyPressed(KeyEvent e)
16.    {
17.        if(e.getKeyCode()==KeyEvent.VK_CONTROL)
18.        {
19.            int R = (int)(Math.random()*100) % 255;
20.            int G = (int)(Math.random()*100) % 255;
21.            int B = (int)(Math.random()*100) % 255;
```

```

22.     Color mycolor = new Color(R,G,B);
23.     this.setBackground(mycolor);
24. }
25.
26. //Empty function
27. public void keyReleased(KeyEvent e)
28. {
29. }
30. //Empty function
31. public void keyTyped(KeyEvent e)
32. {
33. }
34.*/
35.*/
36.<applet code=Change_Background.class width=400 height=400>
37.</applet>
38.*/

```

To compile and run the program use the following commands :

```

>>>javac Change_Background.java
>>>appletviewer Change_Background.java

```

Program Explanation

1. Add **KeyListener** to the applet.
2. When any key is pressed, the function **keyPressed** is called.
3. The key code of Control key is available as **KeyEvent.VK_CONTROL**. Compare it with the key code of the key pressed.
4. If the Control key is pressed, generate three random integer whose value are in the range 0 to 255 and use them as the RGB value to create a color.
5. Set the color created by the random values as the background color of the applet.

advertisement

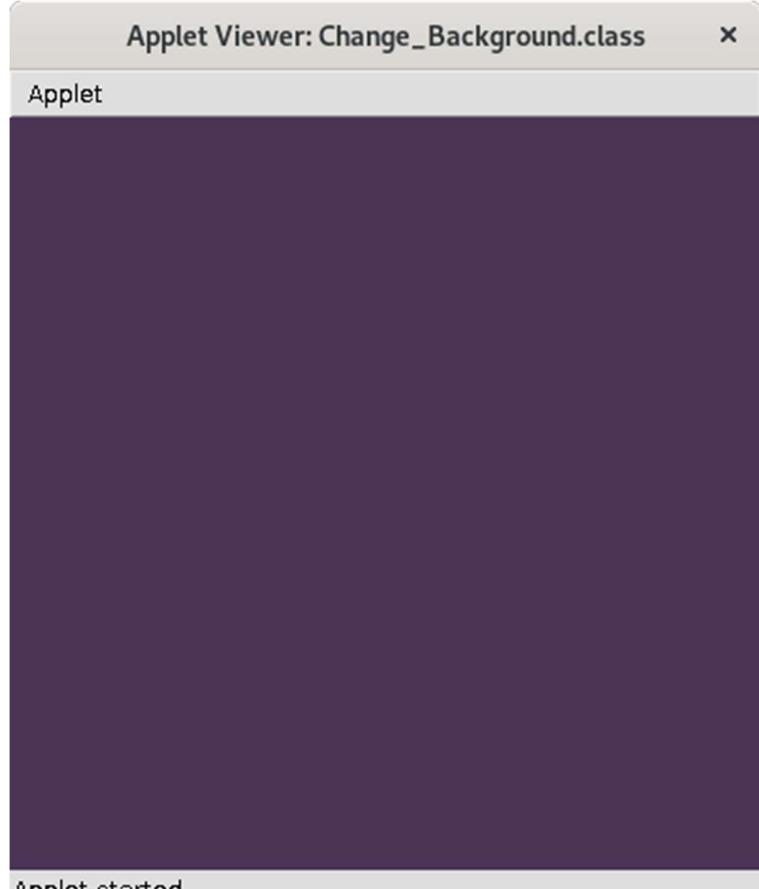
Runtime Test Cases

Here's the run time test cases for changing background color of applet.

Test case 1 – To View the Initial Color of Applet.



Test case 2 – To View the Color of Applet on Click of Control Button.



Applet started.

Test case 3 – To View the Color of Applet on Click of Control Button.



Applet started.

141. Java Program to Play Sound Using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Play Sound Using Applet

Problem Description

We have to write a program in Java such that it creates a music player to pause, play, stop and replay a sound.

Expected Input and Output

The audio file is saved as “audio.wav”

advertisement

Consider buttons are created with the following image icons :

For Pause Button : The icon is saved as **icon_pause.png**



For Play Button : The icon is saved as **icon_play.png**



For Stop Button : The icon is saved as **icon_stop.png**



advertisement

For Replay Button : The icon is saved as **icon_replay.png**



For playing sound using applet, we can have the following 5 different sets of input and output.

1. To Play the Sound : Before the sound is Paused

When the Play button is clicked,
it is expected that the sound is played.

2. To Pause the Sound :

advertisement

When the Pause button is clicked and sound is being played,
it is expected that the sound is paused.

3. To Play the Sound : After the sound is Paused

When the Play button is clicked,
it is expected that the sound is played from the paused position.

4. To Stop the Sound :

advertisement

When the Stop button is clicked,
it is expected that the sound is stopped.

5. To Replay the Sound :

When the Replay button is clicked,
it is expected that the sound is started from the beginning.

Problem Solution

1. Create buttons with icons for the options – pause, play, stop and replay.
2. Load the sound file using **AudioInputStream**.
3. Create a object of class **Clip** to load the sound in a data line.
4. When the button pause is clicked, stop the clip and store the last position of the clip.
5. When the button play is clicked, start the clip from the last position of being stopped. In case the sound is never stopped, play from the beginning.
6. When the button stop is clicked, stop the clip and reset the last position to the beginning.
7. When the button replay is clicked, start the clip from the beginning.

Program/Source Code

Here is source code of the Java Program to play a sound using an applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /* Java Program to Play Sound using Applet */  
2. import java.awt.*;  
3. import java.applet.*;  
4. import javax.swing.*;  
5. import java.awt.event.*;  
6. import javax.sound.sampled.*;  
7. import java.io.File;  
8. public class Music extends Applet implements ActionListener  
9. {  
10.    ImageIcon pause,play,stop,replay;  
11.    long current;  
12.    Clip clip;  
13.    //Initialize the applet  
14.    public void init()  
15.    {  
16.       setBackground(Color.white);  
17.       //Create image icons for the buttons  
18.       pause = new ImageIcon("icon_pause.png");  
19.       play = new ImageIcon("icon_play.png");  
20.       stop = new ImageIcon("icon_stop.png");  
21.       replay = new ImageIcon("icon_replay.png");  
22.    }  
23.    //Add buttons to applet and load the sound  
24.    public void start()  
25.    {  
26.       JButton b_pause = new JButton(pause);  
27.       this.add(b_pause);  
28.       b_pause.addActionListener(this);
```

```

29.
30. JButton b_play = new JButton(play);
31. this.add(b_play);
32. b_play.addActionListener(this);
33.
34. JButton b_stop = new JButton(stop);
35. this.add(b_stop);
36. b_stop.addActionListener(this);
37.
38. JButton b_replay = new JButton(replay);
39. this.add(b_replay);
40. b_replay.addActionListener(this);
41.
42. try
43. {
44.     AudioInputStream audio;
45.     File file = new File ("sound.wav");
46.     audio = AudioSystem.getAudioInputStream(file);
47.     clip = AudioSystem.getClip();
48.     clip.open(audio);
49.     current=0L;
50. }
51. catch(Exception E)
52. {
53.     System.out.println(E.getMessage());
54. }
55. }
56. //Function to perform the selected option
57. public void actionPerformed(ActionEvent e)
58. {
59.     String icon = ((JButton)e.getSource()).getIcon().toString();
60.
61.     if(icon.equals(pause.toString()))
62.     {
63.         current = clip.getMicrosecondPosition();
64.         clip.stop();
65.     }
66.     else if(icon.equals(play.toString()))
67.     {
68.         clip.setMicrosecondPosition(current);
69.         clip.start();
70.     }
71.     else if(icon.equals(stop.toString()))
72.     {
73.         current = 0L;
74.         clip.setMicrosecondPosition(0);
75.         clip.stop();
76.     }
77.     else
78.     {
79.         current = 0L;
80.         clip.setMicrosecondPosition(0);
81.         clip.start();
82.     }
83.
84. }
85.}
86.*/
87.<applet code = Music.java width=750 height=400>
88.</applet>
89.*/

```

To compile and execute the program use the following commands :

```

>>> javac Music.java
>>> appletviewer Music.java

```

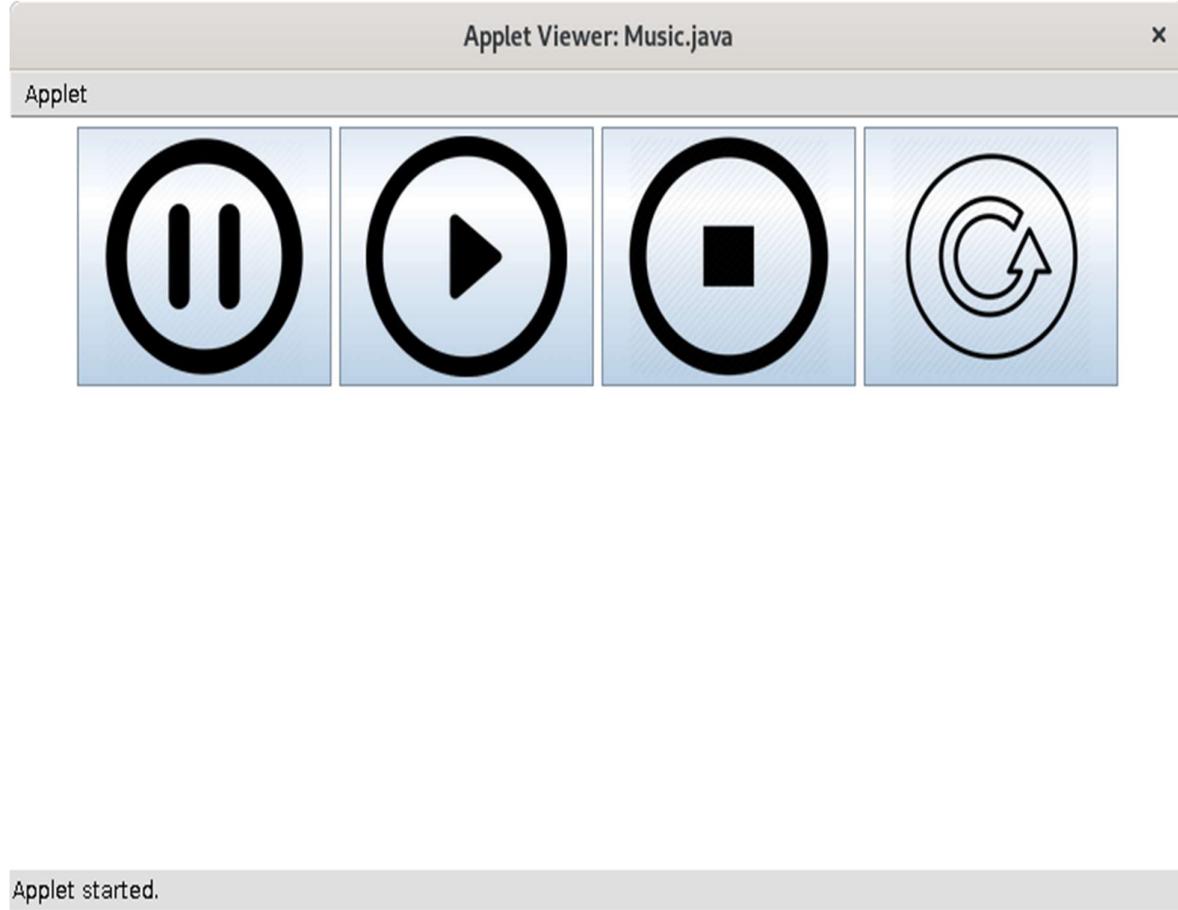
Program Explanation

1. The class **AudioInputStream** is used to load an audio file.
2. To load an audio file to stream, use **AudioSystem.getAudioInputStream(File)**.
3. To convert stream to data line, use **AudioSystem.getClip()**.
4. To get the current position of the clip, use **getMicrosecondPosition** method.
5. To set the position of the clip, use **setMicrosecondPosition(long)** method.
6. To start the audio, use **clip.start()**
7. To stop the audio, use **clip.stop()**

Runtime Test Cases

Here's the run time test cases for playing a sound using an applet.

Test case 1 – To View the Applet



142. Java Program to Create a Transparent Cursor

« [Prev](#)

[Next](#) »

This is a Java Program to Create a Transparent Cursor

Problem Description

We have to write a program in Java such that it creates an applet in which the cursor behaves as a transparent cursor inside an area.

Expected Input and Output

For creating a transparent cursor, we can have the following set of input and output.

advertisement

To View the Transparent Cursor :

When the cursor is moved inside the text area,
it is expected that the cursor becomes transparent.

Problem Solution

1. Create a object of class **Toolkit** and get the default system toolkit.
2. Create an object of class **Image** for the cursor image. For a transparent cursor the image path is empty.
3. Create an object of class **Point**.
4. Create a cursor using the **createCustomCursor** method of the **Cursor** class.
5. Create a text area and add the cursor to the area.

advertisement

Program/Source Code

Here is source code of the Java Program to create a transparent cursor. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to Create a Transparent Cursor using an Applet */
2. import java.applet.*;
3. import java.awt.*;
4. public class My_Cursor extends Applet
5. {
6.     //Function to create a transparent cursor
7.     public void init()
8.     {
9.         setBackground(Color.white);
10.        setLayout(null);
11.        //Create a transparent cursor
12.        Toolkit t = Toolkit.getDefaultToolkit();
13.        Image img = t.getImage("");
14.        Point p = new Point(0,0);
15.        Cursor c = t.createCustomCursor(img,p,"my_cursor");
16.        //Create a area when the cursor is transparent
17.        TextArea area = new TextArea();
18.        area.setBackground(Color.green);
19.        area.setBounds(100,100,300,300);
20.        this.add(area);
21.        //Set transparent cursor to the area
22.        area.setCursor(c);
23.    }
24.}
25.*/
26.<applet code = My_Cursor.class width = 500 height = 500>
27.</applet>
```

28. */

To compile and execute the program use the following commands :

advertisement

```
>>>javac My_Cursor.java  
>>>appletviewer My_Cursor.java
```

Program Explanation

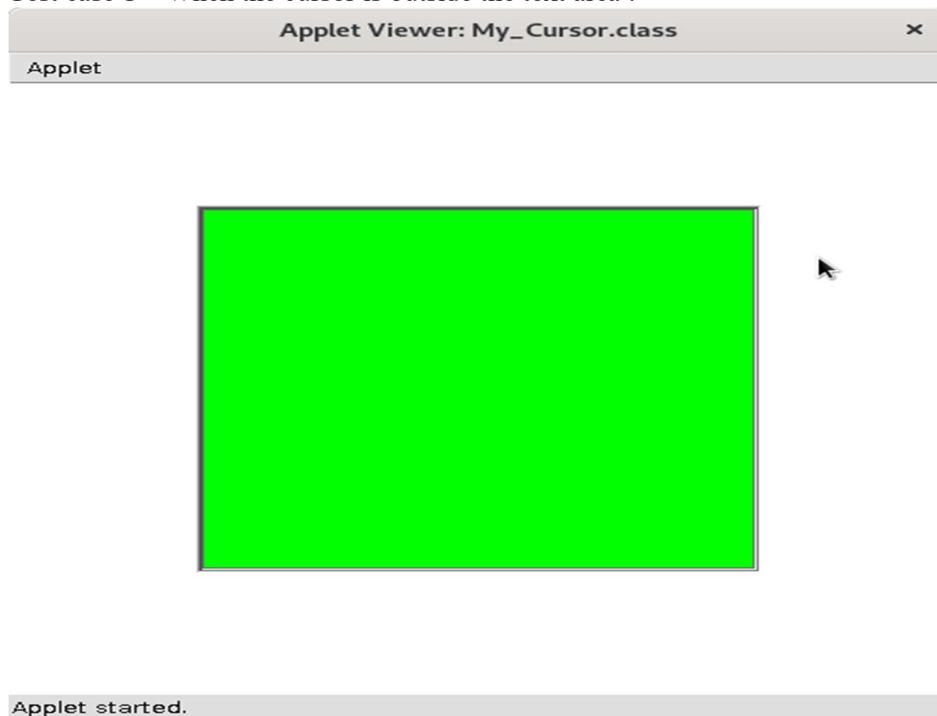
1. The method **createCustomCursor(Cursor Image, Point Hotspot, Cursor Name)** is used to create a cursor.
2. To set the cursor to a component use **setCursor(Cursor)** method.

Runtime Test Cases

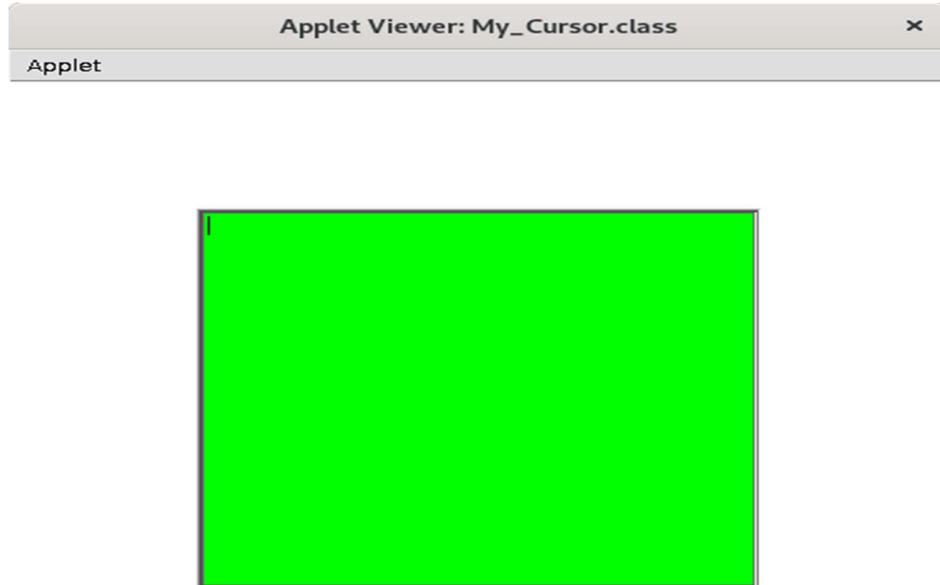
Here's the run time test cases for creating a custom cursor for different input cases.

advertisement

Test case 1 – When the cursor is outside the text area :

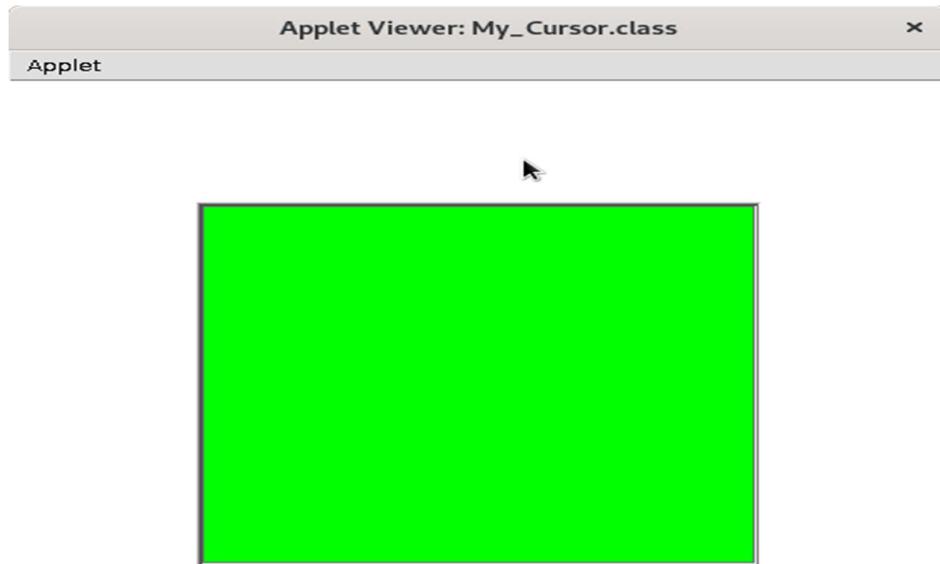


Test case 2 – When the cursor is inside the text area :



Applet started.

Test case 3 – When the cursor is outside the text area :



Applet started.

143. Java Program to Create 2 Labels and Text Fields for Entering Name and Secured Password

[« Prev](#)

[Next »](#)

This is a Java program to create 2 labels and text fields for entering name and secured password.

Problem Description

We have to write a program in Java such that it creates a secured login field containing a name and secured password. When the user clicks view button, the username and password will be displayed.

Expected Input and Output

For creating a secured login field, we can have the following set of input and output.

advertisement

To display the username and password :

If the name 'thomas' and password 'thomas@123' are typed in the respective field, the expected output is "Username = thomas Password = thomas@123"

Problem Solution

1. Create a frame, two labels for 'Name' and 'Password', a text field for name, a password field for password, a text field for output and a button to view the output.
2. Position the fields and button properly on the frame.
3. Associate **ActionListener** with the button.
4. When the button is clicked, display the username and password in the output text field.

advertisement

Program/Source Code

Here is source code of the Java Program to create a secured login field. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. /*Java Program to create a secured login field*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Login implements ActionListener
6. {
7.     static JTextField name;
8.     static JPasswordField pswd;
9.     static JButton view;
10.    static JTextField text;
11.    //Driver function
12.    public static void main(String args[])
13.    {
14.        //Create a frame
15.        JFrame frame=new JFrame("Login");
16.        frame.setSize(400,400);
17.        frame.setLayout(null);
18.        frame.setBackground(Color.white);
19.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20.        //Create two labels
21.        JLabel l_name=new JLabel("Name :");
22.        JLabel l_pswd=new JLabel("Password :");
23.        l_name.setBounds(50,50,50,50);
24.        l_pswd.setBounds(20,100,80,50);
25.        frame.add(l_name);
```

```

26.         frame.add(l_pswd);
27.         //Create two text fields for name and password
28.         name=new JTextField();
29.         pswd=new JPasswordField();
30.         name.setBounds(100,50,250,50);
31.         pswd.setBounds(100,100,250,50);
32.         frame.add(name);
33.         frame.add(pswd);
34.         //Create a button
35.         view=new JButton("View");
36.         view.setBounds(150,200,100,50);
37.         frame.add(view);
38.         //Create a text field to display the username and password
39.         text=new JTextField();
40.         text.setBounds(0,250,400,50);
41.         frame.add(text);
42.         //Create an object
43.         Login obj=new Login();
44.         //Associate ActionListener with the button
45.         view.addActionListener(obj);
46.         //Display the frame
47.         frame.setVisible(true);
48.     }
49.     //function to display the username and password typed
50.     public void actionPerformed(ActionEvent e)
51.     {
52.         String pass=String.valueOf(pswd.getPassword());
53.         text.setText("Username = "+name.getText());
54.         text.setText(text.getText()+" Password = "+pass);
55.     }
56.

```

Program Explanation

1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(400,400);** is used to set the width and height of the frame.
2. **frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);** is a default close operation for the frame.
3. Create 2 labels “Name” and “Password” by using **JLabel l_name=new JLabel(“Name :”);** and **JLabel l_pswd=new JLabel(“Password :”);**. Once label is created add it to the frame by using **frame.add();**.
4. By using **setBounds()** values we can set the position of label names.
5. Create two text fields for name and password with **setBounds(int x, int y, int width, int height)**, where ‘x’ & ‘y’ is used to set the position of top-left corner. ‘int width’ & ‘int height’ is used to set the width and height of the exit button. Once fields are created add them to the frame by using **frame.add();**.
 - name=new JTextField();** –> Creates a textfield for name.
 - pswd=new JPasswordField();** –> Creates a password field for secured password.
 - name.setBounds(100,50,250,50);** –> Sets the position of name textfield.
 - pswd.setBounds(100,100,250,50);** –> Sets the position of password field.
6. Create a view button with a **setBounds()** values and add it to the frame by using **frame.add();**.
 - view=new JButton(“View”);** –> Used to create the view button.
 - view.setBounds(150,200,100,50);** –> Sets the position of button.
 - frame.add(view);** –> It will add the button to the frame.
7. Create one more text field for output to display the username and password.
8. **Login obj=new Login();** is used to create the object. Once object is created associate **ActionListener** with the button.
9. **frame.setVisible(true);** is used to display the frame.
10. **String pass=String.valueOf(pswd.getPassword());** –> **getPassword** returns a **char[]** of the string typed in the password field. It is converted to **String** format by using function **String.valueOf**.
11. When the View button is clicked, it display the username and password in the output text field.
 - text.setText(“Username = “+name.getText());** –> It displays the name in output text field
 - text.setText(text.getText()+" Password = “+pass);** –> It displays the password in output text field.

Runtime Test Cases

Here's the run time test case for getting username and password from a secured login field.

advertisement

Test case – Here's the runtime output to get the username and password. For example, if user enters Name as “thomas” and Password as “thomas@123” in the text field. After clicking view button it should display “Username = thomas Password = thomas@123”.

The screenshot shows a "Login" application window. At the top, there is a title bar with the word "Login" and a close button (X). Below the title bar, there are two input fields: one for "Name" containing "thomas" and one for "Password" containing ".....". Below these fields is a blue "View" button. At the bottom of the window, there is a status message box displaying "Username = thomas Password = thomas@123".

144. Java Program to Create a List Box to Select Multiple Items and Display it in the Frame

[« Prev](#)

[Next »](#)

This is a Java program to create a list box to select multiple items and display it in the frame.

Problem Description

We have to write a program in Java such that it creates a list box (C, C++, Java, Python and R) from which the user can select one or multiple items, and the selected items are displayed in the frame.

Expected Input and Output

For displaying the items selected from a list box, we can have the following sets of input and output. Consider the languages C, C++, Java, Python and R are in the list box.

advertisement

1. To display the list box:

If the program is executed,
then it is expected that a list box is displayed with the below languages.
C, C++, Java, Python and R

2. To select one item:

If any one item, say Java is selected from the list (C, C++, Java, Python and R),
then the expected output is "Items Selected : Java".

3. To select three items:

advertisement

If any three items, say C++, Java, R are selected from the list (C, C++, Java, Python and R),
then the expected output is "Items Selected : C++, Java, R".

4. To select all items:

If all items, say C, C++, Java, Python, R are selected from the list (C, C++, Java, Python and R),
then the expected output is "Items Selected : C, C++, Java, Python, R".

Problem Solution

1. Create a frame and a list of type **String** with the languages, and a label.
2. Add **ListSelectionListener** to the list. The **ListSelectionListener** defines method **public void valueChanged(ListSelectionEvent)**.
3. Get the array of indices selected and get the corresponding items.
4. Display the items selected.

advertisement

Program/Source Code

Here is source code of the Java Program to display the items selected from a list box. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. /*Java Program to display items selected from a list box*/
2. import javax.swing.*;
3. import java.awt.*;
4. import javax.swing.event.*;
5. class List_Box implements ListSelectionListener
6. {
7.     static JList<String> list;
8.     static JLabel label;
9.     //Driver function
```

```

10. public static void main(String args[])
11. {
12.     //Create a frame
13.     JFrame frame=new JFrame("List Box");
14.     frame.setSize(500,500);
15.     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.     frame.setLayout(null);
17.     frame.getContentPane().setBackground(Color.white);
18.     //Create a list
19.     String lang[]={ "C", "C++", "Java", "Python", "R" };
20.     list=new JList<String>(lang);
21.     list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
22.     list.setBounds(150,100,150,150);
23.     frame.add(list);
24.     //Create a label
25.     label=new JLabel("Select items from list");
26.     label.setBounds(0,0,500,50);
27.     frame.add(label);
28.     //Create an object
29.     List_Box obj=new List_Box();
30.     //Add ListSelectionListener to list
31.     list.addListSelectionListener(obj);
32.     //Display the frame
33.     frame.setVisible(true);
34. }
35. //Function to display the items selected
36. public void valueChanged(ListSelectionEvent e)
37. {
38.     //Get index of items selected
39.     int index[]=list.getSelectedIndices();
40.     //Get the items selected from their indices
41.     String str="";
42.     for(int i=0;i<index.length;i++)
43.         str=str+list.getModel().getElementAt(index[i])+", ";
44.     str=str.replaceAll(", $", "");
45.     //Change label to items selected
46.     label.setText("Items Selected : "+str);
47. }
48.

```

Program Explanation

1. Create a frame with background color & specific size. `frame.setBackground(Color.white);` is used to create the frame background color as white. `frame.setSize(500,500);` is used to set the width and height of the frame.

2. `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` is a default close operation for the frame.

3. Create a list using `String lang[]` and add it to the frame by using `frame.add()`.

`String lang[]={ "C", "C++", "Java", "Python", "R" };` → array of languages (C, C++, Java, Python and R) is added to the list.

`list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);` → It allows user to select multiple items in the list.

`list.setBounds(150,100,150,150);` → Sets the position of list.

`frame.add(list);` → It will add the list to the frame.

4. `list=new JList(lang);` → A List of type string is created using `JList<E>` where E is the type of data with the list will have.

5. Create a label with a `setBounds()` values and add it to the frame. This label is used for output to display a selected items from the list. By default label name will be **Select items from list**.

6. `List_Box obj=new List_Box();` is used to create the object. Once object is created associate `ListSelectionListener` with the list.

7. `ListSelectionListener` of the `javax.swing.event` package is associated with the list created.

8. In method `valueChanged` the indices of selected items is obtained using function `getSelectedIndices()`, which returns an array of integers.

9. The item corresponding to the index is obtained using function `getElementAt(int)`, which returns the item name.

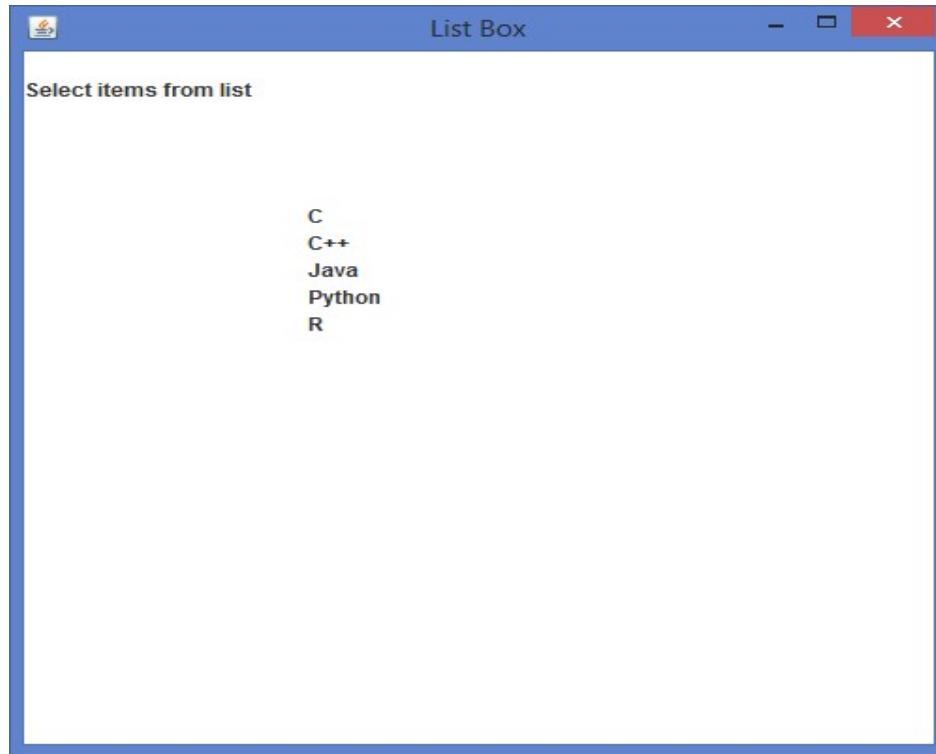
10. `label.setText("Items Selected : "+str);` -> It will change the label name from **Select items from list** to **Items Selected :** and also displays the selected items from the list.

advertisement

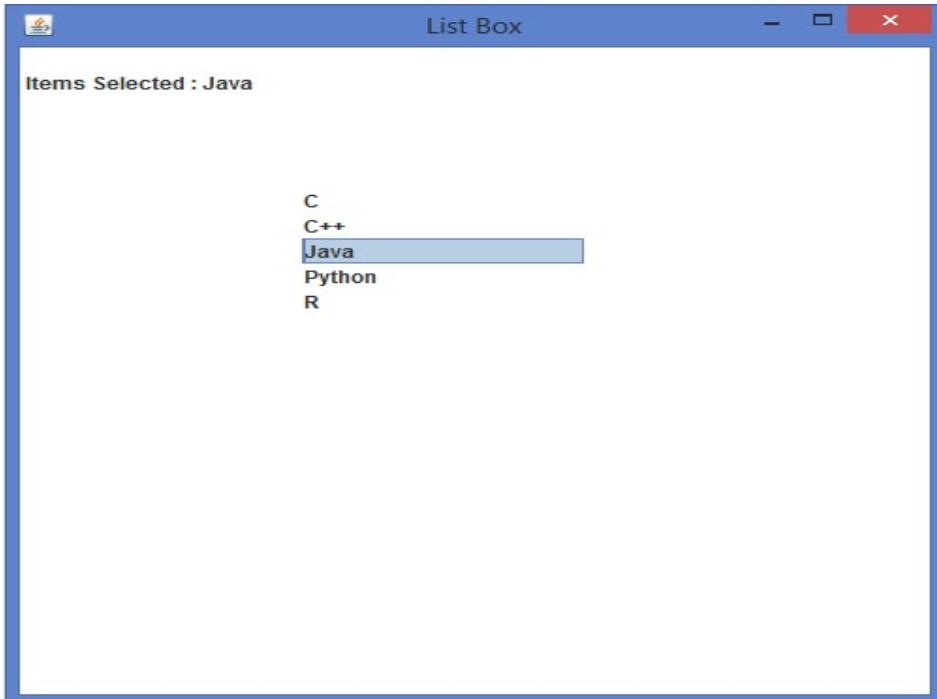
Runtime Test Cases

Here's the run time test case for displaying items selected from a list.

Test case 1 – Here's the runtime output to view the list. When the program is executed, then it displays a list box with the following languages (C, C++, Java, Python and R).

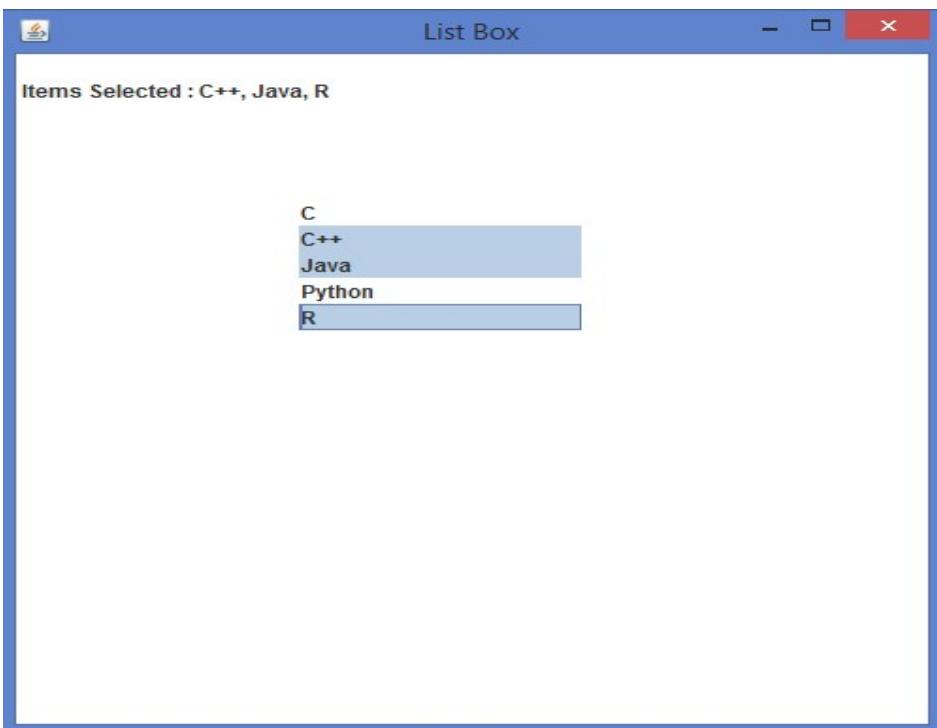


Test case 2 – Here's the runtime output to select any one item. For example, if the language Java is selected from the list (C, C++, Java, Python, R), then the program displays “Items Selected : Java”.

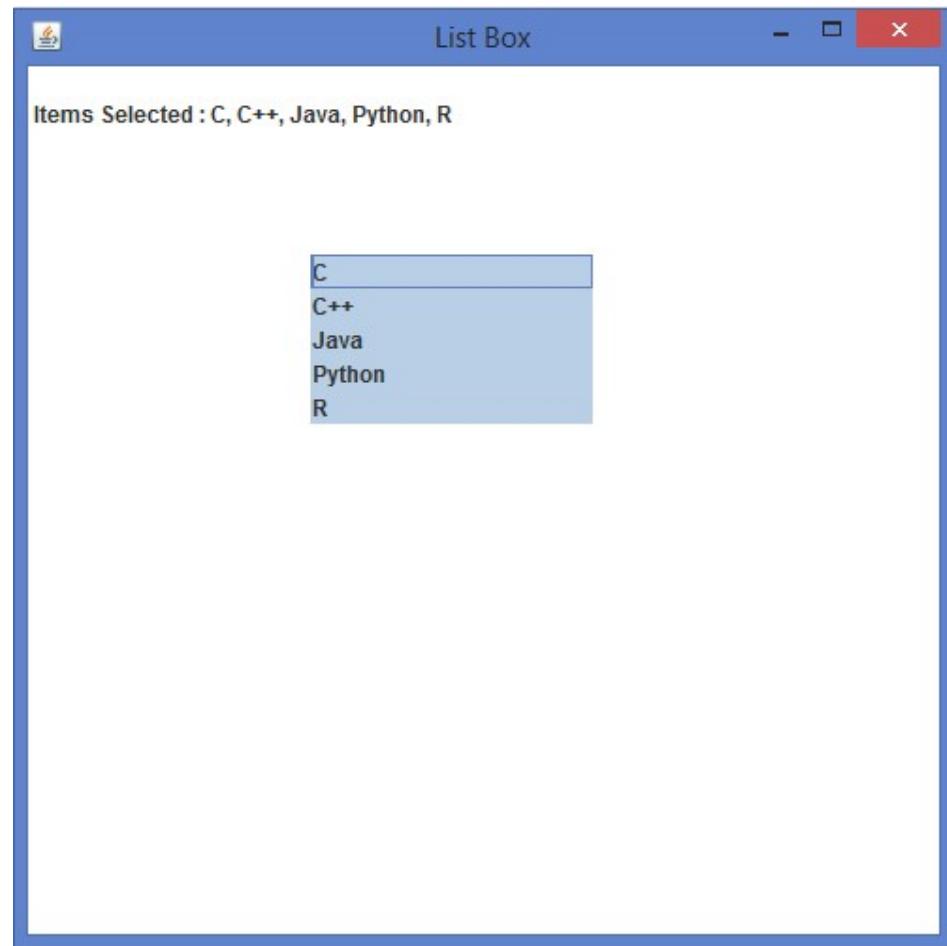


advertisement

Test case 3 – Here's the runtime output to select any three items. For example, if the language C++, Java, R is selected from the list (C, C++, Java, Python, R), then the program displays “Items Selected : C++, Java, R”.



Test case 4 – Here's the runtime output to select all items. For example, if all the languages are selected from the list (C, C++, Java, Python, R), then the program displays “Items Selected : C, C++, Java, Python, R”.



145. Java Program to Create 2 Frames and Switch Between the Frames using Buttons

[« Prev](#)

[Next »](#)

This is a Java program to create 2 frames and switch between the frames using buttons.

Problem Description

We have to write a program in Java such that it creates 2 frames: Frame 1 & Frame 2. Frame 1 consists of two buttons: Next and Close. Frame 2 consists of a Back button. Switching between the frames is done using that buttons.

Expected Input and Output

For switching between frames using buttons, we can have the following sets of input and output.

advertisement

1. To View Frame 1 :

If the program is being executed,
then it is expected that it will create and displays Frame 1 with Next and Close buttons.

2. To Create Frame 2 using Next Button :

If the button Next is clicked on Frame 1,
then it is expected that Frame 2 is created.

3. To Close Frame 2 using Back Button :

advertisement

If the button Back is clicked on Frame 2,
then it is expected that Frame 2 is closed and Frame 1 remains activated.

4. To Close Frame 1 using Close Button :

If the button Close is clicked on Frame 1,
then it is expected that Frame 1 is closed.

Problem Solution

1. Create **Frame 1** with buttons **Next** and **Close**.
2. Associate **ActionListener** with the buttons.
3. If button **Next** is clicked, create **Frame 2** with **Back** button.
4. If button **Close** is clicked, close Frame 1.
5. If button **Back** is clicked, close Frame 2.

advertisement

Program/Source Code

Here is source code of the Java Program to create 2 Frames and switch between them using buttons. The program is successfully compiled and tested using javac compiler on Fedora 30.

```
1. /*Java Program to switch between frames using buttons*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Switch_Frame implements ActionListener
6. {
7.     static JFrame frame1;
8.     static JFrame frame2;
9.     static JButton next;
10.    static JButton close;
```

```

11. static JButton back;
12. //Driver function
13. public static void main(String args[])
14. {
15.     //Create frame 1
16.     frame1 = new JFrame("Frame 1");
17.     frame1.setSize(250,250);
18.     frame1.setLayout(null);
19.     frame1.setBackground(Color.white);
20.     //Create next and close buttons
21.     next = new JButton("Next");
22.     close = new JButton("Close");
23.     next.setBounds(75,50,100,50);
24.     close.setBounds(75,150,100,50);
25.     //Add the buttons to frame 1
26.     frame1.add(next);
27.     frame1.add(close);
28.     //Create an object
29.     Switch_Frame obj=new Switch_Frame();
30.     //Associate ActionListener with the buttons
31.     next.addActionListener(obj);
32.     close.addActionListener(obj);
33.     //Display frame 1
34.     frame1.setVisible(true);
35. }
36. //Function to perform actions related to button clicked
37. public void actionPerformed(ActionEvent e)
38. {
39.     String button=e.getActionCommand();
40.     if(button.equals("Next"))
41.     {
42.         create_frame2();
43.     }
44.     if(button.equals("Close"))
45.     {
46.         frame1.dispose();
47.     }
48.     if(button.equals("Back"))
49.     {
50.         frame2.dispose();
51.     }
52. }
53. //function to create Frame 2
54. public static void create_frame2()
55. {
56.     //Create frame 2
57.     frame2 = new JFrame("Frame 2");
58.     frame2.setSize(250,250);
59.     frame2.setLayout(null);
60.     frame2.setBackground(Color.white);
61.     //Create back button
62.     back = new JButton("Back");
63.     back.setBounds(75,100,100,50);
64.     //Add the button to frame 2
65.     frame2.add(back);
66.     //Create an object
67.     Switch_Frame obj=new Switch_Frame();
68.     //Associate ActionListener with the buttons
69.     back.addActionListener(obj);
70.     //Display frame 2
71.     frame2.setVisible(true);
72. }
73. }
```

Program Explanation

1. Create a frame 1 with background color & specific size. `frame1.setBackground(Color.white);` is used to create the frame background color as white. `frame1.setSize(250,250);` is used to set the width and height of the frame.

2. Create a next and close buttons with a `setBounds()` values and add it to the frame.

`next = new JButton("Next");` -> Used to create the next button.

`close = new JButton("Close");` -> Used to create the close button.

`next.setBounds(75,50,100,50);` -> Sets the position of next button.

`close.setBounds(75,150,100,50);` -> Sets the position of close button.

`frame1.add(next);` -> It will add the next button to the frame 1.

`frame1.add(close);` -> It will add the close button to the frame 1.

3. `Switch_Frame obj=new Switch_Frame();` is used to create the object. Once object is created associate `ActionListener` with the next and close buttons as follows.

`next.addActionListener(obj);` -> Associates ActionListener with the next button.

`close.addActionListener(obj);` -> Associates ActionListener with the close button.

advertisement

4. `frame1.setVisible(true);` is used to display the frame 1.

5. Create a frame 2 with background color & specific size. `frame2.setBackground(Color.white);` is used to create the frame background color as white. `frame2.setSize(250,250);` is used to set the width and height of the frame.

6. Create a next and back button with a `setBounds()` values and add it to the frame by using `frame.add();`

`back = new JButton("Back");` -> Used to create the back button.

`back.setBounds(75,100,100,50);` -> Sets the position of back button.

`frame2.add(back)` -> It will add the back button to the frame 2.

7. `Switch_Frame obj=new Switch_Frame();` is used to create the object. Once object is created associate `ActionListener` with the back button.

8. `frame2.setVisible(true);` is used to display the frame 2.

9. `ActionEvent` class is used to perform component defined action. In method `actionPerformed` the following actions will be performed.

i. Initially, **Frame 1** is created with **Next** and **Close** buttons.

ii. If **Next** button is clicked on **Frame 1**, it will create **Frame 2** with **Back** button.

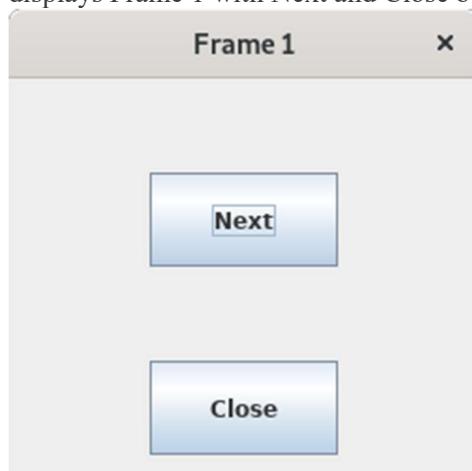
iii. If **Back** button is clicked on **Frame 2**, it will close **close Frame 2**.

iv. If **Close** button is clicked on **Frame 1**, it will close **Frame 1**.

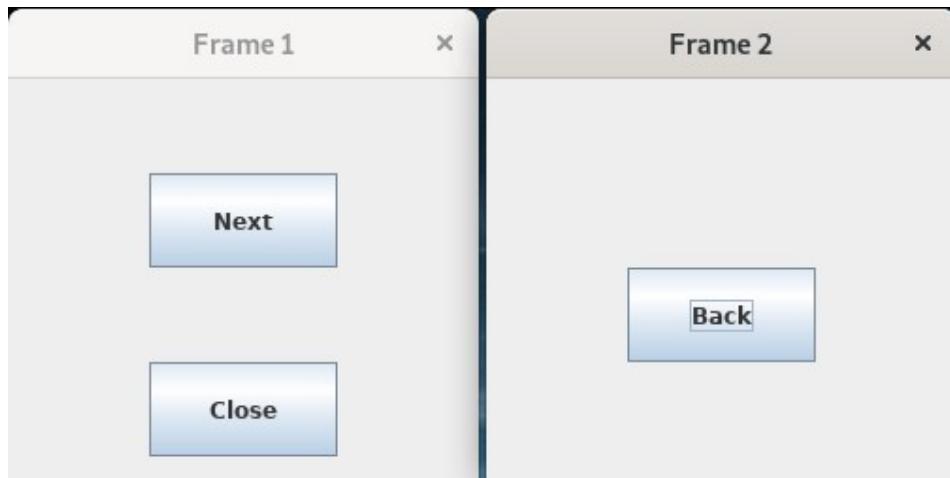
Runtime Test Cases

Here's the run time test case for switching between frames using buttons.

Test case 1 – Here's the runtime output to view Frame 1. When the program is executed, then it will create and displays Frame 1 with Next and Close buttons.

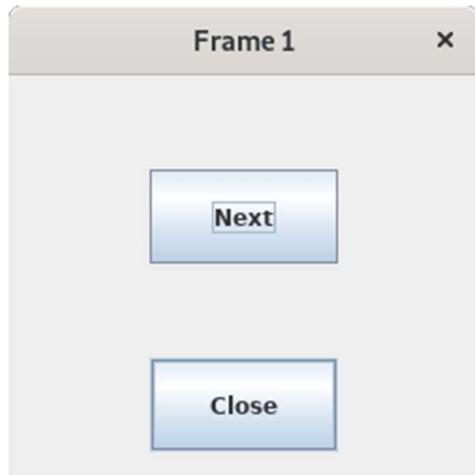


Test case 2 – Here's the runtime output to create Frame 2. Suppose if user clicks on Next button on Frame 1, then it will create Frame 2 with Back button.



advertisement

Test case 3 – Here's the runtime output to close Frame 2. Suppose if user clicks on Back button on Frame 2, then it will close the Frame 2 but Frame 1 remains activated.



146. Java Program to Display Some Text in the Frame with the Help of a Label

[« Prev](#)

[Next »](#)

This is a Java program to display some text in the frame with the help of a label.

Problem Description

We have to write a program in Java such that it displays some text in a frame using JLabel.

Expected Input and Output

To display text using a label, we have the following set of input and output.

advertisement

To display a text:

At the execution of the program, it is expected that a frame appears with some text written with the help of a label.

Text is “This text is written using a label”

Problem Solution

1. Create a frame and a label.
2. Add the label to the frame.
3. Write some text to the label.
4. Display the frame.

Program/Source Code

Here is source code of the Java Program to display text in a frame using a label. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to display text using label*/
2. import javax.swing.*;
3. import java.awt.*;
4. class Text_Label
5. {
6.     //Driver function
7.     public static void main(String args[])
8.     {
9.         //Create a frame
10.        JFrame frame=new JFrame("Text using Label");
11.        frame.setSize(500,500);
12.        frame.setBackground(Color.white);
13.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14.        frame.setLayout(null);
15.        //Create a Label
16.        JLabel label=new JLabel();
17.        label.setBounds(0,100,500,50);
18.        frame.add(label);
19.        //Write text to the label
20.        String str="This text is written using a label";
21.        label.setText(str);
22.        //Display the frame
23.        frame.setVisible(true);
24. }
```

Program Explanation

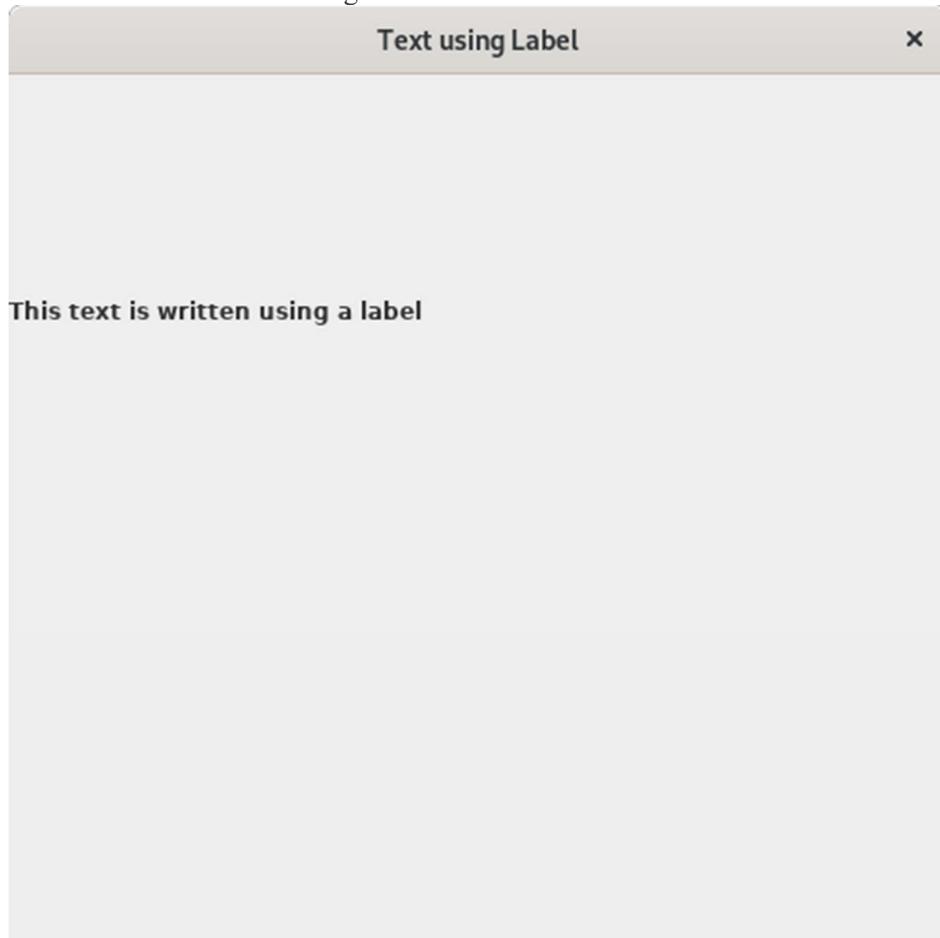
1. Create a frame with background color & specific size. `frame.setBackground(Color.white);` is used to create the frame background color as white. `frame.setSize(500,500);` is used to set the width and height of the frame.
2. `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` is a default close operation for the frame.
3. Create a label with a `setBounds()` values and add it to the frame by using `frame.add();`
`JLabel label=new JLabel();` -> Used to create the label.
`label.setBounds(0,100,500,50);` -> Sets the position of label.
`frame.add(label);` -> It will add the label to the frame.
4. `String str="This text is written using a label";` -> It is used to write text to the label. Text is “**This text is written using a label**”
5. `label.setText(str);` -> It will add the text to the label.
6. `frame.setVisible(true);` is used to display the frame. The frame will display with the text “**This text is written using a label**” which is added to the label.

advertisement

Runtime Test Cases

Here's the run time test case for displaying text in a frame using a label.

Test case – Here's the runtime output to display a text. Once the program is executed, it displays the frame with text “This text is written using a label”.



147. Java Program to Create Push Buttons and Draw Different Borders Around the Buttons

[« Prev](#)

[Next »](#)

This is a Java program to create push buttons and draw different borders around the buttons.

Problem Description

We have to write a program in Java such that it creates a frame containing five Push Buttons and different types of border are drawn around it.

Expected Input and Output

To draw different borders around the buttons, we have the following set of input and output.

advertisement

To display the borders:

At the execution of the program, it is expected that a frame appears with five push borders having different types of border around them.

Button 1 – LineBorder with black color and 2px size.

Button 2 – LineBorder with black color and 10px size.

Button 3 – Etched Raised Border with highlighting black color and white as the shadow color.

Button 4 – Etched Lowered Border with highlighting black color and white as the shadow color.

Button 5 – Bevel Raised Border with highlighting black color and white as the shadow color.

Problem Solution

1. Create a frame and push buttons.
2. Create different types of borders as follows :-
 - a) **Thin Border:** Create a border of class **LineBorder** of color black and size 2px.
 - b) **Thick Border:** Create a border of class **LineBorder** of color black and size 10px.
 - c) **Etched Raised Border:** Create a border of class **EtchedBorder** of type **RAISED** having highlight color black and shadow color white.
 - d) **Etched Lowered Border:** Create a border of class **EtchedBorder** of type **LOWERED** having highlight color black and shadow color white.
 - e) **Bevel Raised Border:** Create a border of class **BevelBorder** of type **RAISED** having highlight color black and shadow color white.
3. Add the different borders to buttons.
4. Display the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to draw different borders around buttons. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to draw borders around Push Button*/
2. import javax.swing.*;
3. import java.awt.*;
4. import javax.swing.border.*;
5. class Button_Border
6. {
7.     //Driver function
8.     public static void main(String args[])
9.     {
10.        //Create a frame
```

```

11. JFrame frame=new JFrame("Borders around Buttons");
12. frame.setSize(500,500);
13. frame.setLayout(null);
14. frame.setBackground(Color.white);
15. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16. //Create push buttons
17. JButton[] button=new JButton[5];
18. for(int i=0;i<5;i++)
19. {
20.     button[i]=new JButton("Button "+(i+1));
21.     button[i].setBounds(210,i*75,100,50);
22.     frame.add(button[i]);
23. }
24. //Create different borders
25. LineBorder thin,thick;
26. EtchedBorder raised,low;
27. BevelBorder bevel;
28. thin=new LineBorder(Color.black,2);
29. thick=new LineBorder(Color.black,10);
30. raised=new EtchedBorder(EtchedBorder.RAISED,Color.black,Color.white);
31. low=new EtchedBorder(EtchedBorder.LOWERED,Color.black,Color.white);
32. bevel=new BevelBorder(BevelBorder.RAISED,Color.black,Color.white);
33. //Add borders to buttons
34. button[0].setBorder(thin);
35. button[1].setBorder(thick);
36. button[2].setBorder(raised);
37. button[3].setBorder(low);
38. button[4].setBorder(bevel);
39. //Display frame
40. frame.setVisible(true);
41. }
42.

```

Program Explanation

1. Create a frame with background color & specific size. **frame.setBackground(Color.white);** is used to create the frame background color as white. **frame.setSize(500,500);** is used to set the width and height of the frame.
2. **frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);** is a default close operation for the frame.
3. Create 5 push buttons using for loop and set bound values. Once push buttons are created add them to the frame by using **frame.add();**
4. Create a different borders for each button using different border types like **LineBorder (thin, thick)**, **EtchedBorder (raised, low)** & **BevelBorder**.
5. **thin=new LineBorder(Color.black,2);** → It is used to create a border for **LineBorder** class with black color and 2px size. Thin Border is used for Button 1.
6. **thick=new LineBorder(Color.black,10);** → It is used to create a border for **LineBorder** class with black color and 10px size. Thick Border is used for Button 2.
7. **raised=new EtchedBorder(EtchedBorder.RAISED,Color.black,Color.white);** → It is used to create a border for **EtchedBorder** class with **RAISED** type which is having black color highlighting and white color shadow. Etched Raised Border is used for Button 3.

advertisement

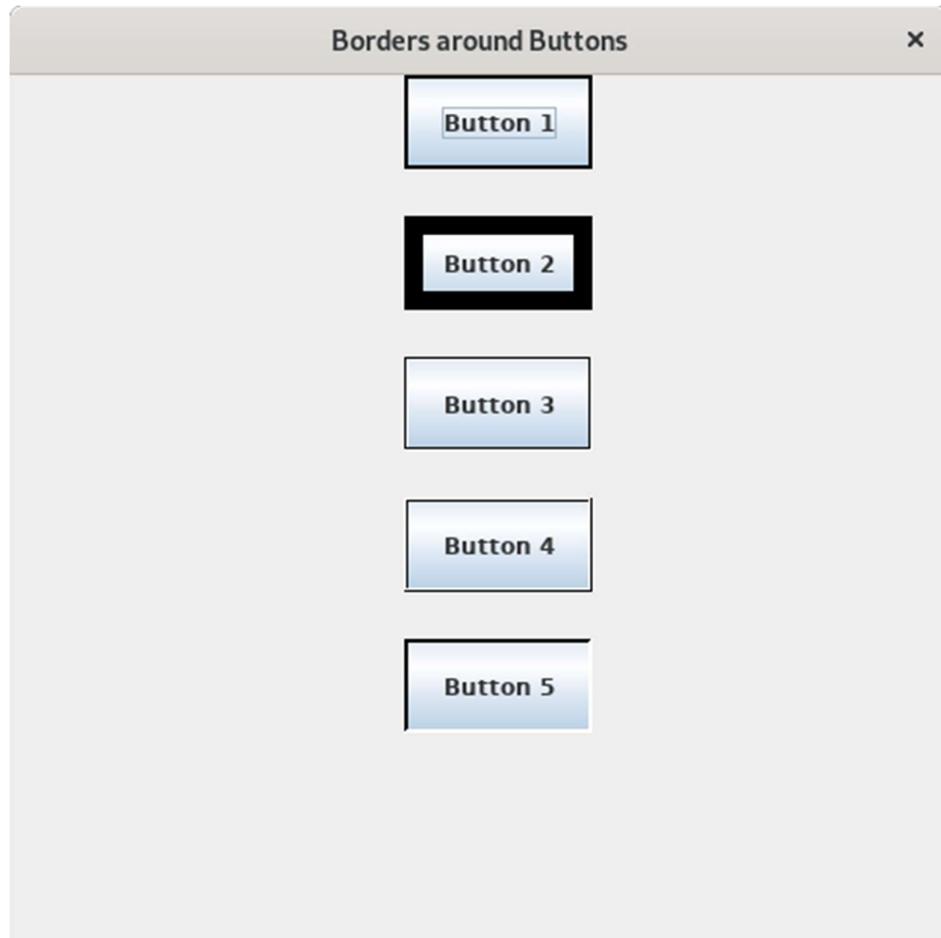
8. **low=new EtchedBorder(EtchedBorder.LOWERED,Color.black,Color.white);** → It is used to create a border for **EtchedBorder** class with **LOWERED** type which is having black color highlighting and white color shadow. Etched LOWERED Border is used for Button 4.
 9. **bevel=new BevelBorder(BevelBorder.RAISED,Color.black,Color.white);** → It is used to create a border for **BevelBorder** class with **RAISED** type which is having black color highlighting and white color shadow. Bevel Raised Border is used for Button 5.
 10. Once borders creation is done. Add borders to buttons as follows:
- button[0].setBorder(thin);** → Thin Border is added for Button 1
 - button[1].setBorder(thick);** → Thick Border is added for Button 2.
 - button[2].setBorder(raised);** → Etched Raised Border is added for Button 3.
 - button[3].setBorder(low);** → Etched LOWERED Border is added for Button 4.
 - button[4].setBorder(bevel);** → Bevel Raised Border is added for Button 5.

11. `frame.setVisible(true);` -> It is used to display the frame.
advertisement

Runtime Test Cases

Here's the run time test case to draw different borders around buttons.

Test case – Here's the runtime output to display the borders around the buttons.
Button 1 displays Thin LineBorder with black color and 2px size.
Button 2 displays Thick LineBorder with black color and 10px size.
Button 3 displays Etched Raised Border with black color highlighting and white as the shadow color.
Button 4 displays Etched Lowered Border with black color highlighting and white as the shadow color.
Button 5 displays Bevel Raised Border with black color highlighting and white as the shadow color.



148. Java Program to Create a Button and Display Image in the Frame when Clicked

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Button and Display Image in the Frame when Clicked.

Problem Description

We have to write a program in Java such that it creates a frame with a button. When the button is clicked an image is displayed in the frame.

Expected Input and Output

To display image on click of button, we have the following set of input and output.

advertisement

To Display the Image on Click:

If an image "logo.jpeg" is available in the present working directory, then it is expected that the frame displays the image when user clicks on display button.

Sample Image :



advertisement

Problem Solution

1. Create a frame with a button.
2. Display the frame.
3. When the button is clicked, create an icon of the image using class **ImageIcon**.
4. Add the icon to a label and then add the label to the frame.

Program/Source Code

Here is source code of the Java Program to display image in frame when button is clicked. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Display Image when Button is Clicked*/
2. import javax.swing.*;
3. import javax.swing.ImageIcon.*;
4. import java.awt.*;
```

```

5. import java.awt.event.*;
6. class Button_Image implements ActionListener
7. {
8.     static JFrame frame;
9.     //Driver function
10.    public static void main(String args[])
11.    {
12.        //Create a frame
13.        frame=new JFrame("Image on Click");
14.        frame.setSize(500,500);
15.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.        frame.getContentPane().setBackground(Color.white);
17.        frame.setLayout(new FlowLayout());
18.        //Create a button
19.        JButton button=new JButton("Display");
20.        frame.add(button);
21.        //Create an object
22.        Button_Image obj=new Button_Image();
23.        //Associate ActionListener with button
24.        button.addActionListener(obj);
25.        //Display the frame
26.        frame.setVisible(true);
27.    }
28.    //Function to display image
29.    public void actionPerformed(ActionEvent e)
30.    {
31.        //Display Image
32.        ImageIcon icon=new ImageIcon("logo.jpeg");
33.        JLabel label=new JLabel(icon);
34.        frame.add(label);
35.        frame.pack();
36.        frame.setSize(500,500);
37.    }
38.}

```

Program Explanation

1. A frame with a button is created.
2. When the button is clicked, an icon of the image is created using class **ImageIcon**. The file name of the image is given as a parameter. It is not required to mention the file path is the icon is in the present working directory. Otherwise, complete file path is required.
3. Add the icon to a label and add the label to the frame.
4. Resize the frame after adding the image to the frame as we are using the **pack** function which may change the size of frame.

advertisement

Runtime Test Cases

Here's the run time test cases to display an image when a button is clicked.

Test case 1 – To display the frame with the button.



advertisement

Test case 2 – To display the icon in the frame.



149. Java Program to Create Check Boxes and Radio Buttons and Display the Selected in a Text Area

[« Prev](#)

[Next »](#)

This is a Java Program to Create Check Boxes and Radio Buttons and Display the Selected in a Text Area

Problem Description

We have to write a program in Java such that it creates 3 Radio Buttons and 3 Check Boxes and displays the selected using two labels.

Expected Input and Output

To display the Radio Buttons and Check Boxes selected, we have the following sets of input and output.

advertisement

1. When Radio Buttons are Selected:

Suppose there are 3 Radio Buttons namely Button 1, Button 2 and Button 3

If we select all 3 Radio Buttons,

then the expected output is "Button(s) Selected : Button 1, Button 2, Button 3"

2. When Check Boxes are Selected:

Suppose there are 3 Radio Buttons & 3 Check Boxes

Button 1, Button 2 and Button 3

Checkbox 1, Checkbox 2 and Checkbox 3

If we select all 3 Radio Buttons & 3 Check Boxes,

then the expected output is

"Button(s) Selected : Button 1, Button 2, Button 3"

"CheckBox(s) Selected : Checkbox 1, Checkbox 2, Checkbox 3"

3. When Radio Buttons are De-Selected:

advertisement

Suppose there are 3 Radio Buttons & 3 Check Boxes

Button 1, Button 2 and Button 3

Checkbox 1, Checkbox 2 and Checkbox 3

If we select 2 Radio Buttons (Button 1 & Button 3) & 3 Check Boxes,

then the expected output is

"Button(s) Selected : Button 1, Button 3"

"CheckBox(s) Selected : Checkbox 1, Checkbox 2, Checkbox 3"

4. When Check Boxes are De-Selected:

Suppose there are 3 Radio Buttons & 3 Check Boxes

Button 1, Button 2 and Button 3

Checkbox 1, Checkbox 2 and Checkbox 3

If we select 2 Radio Buttons (Button 1 & Button 3) & 1 Check Box (Checkbox 2),

then the expected output is

"Button(s) Selected : Button 1, Button 3"

"CheckBox(s) Selected : Checkbox 2"

Problem Solution

1. Create a frame with three Radio Buttons and three Check Boxes respectively.
2. Add **ActionListener** with the Radio Buttons and **ItemListener** with the Check Boxes.
3. Display the list of button(s) selected in label **text1**.
4. Display the list of CheckBox(s) selected in label **text2**.

advertisement

Program/Source Code

Here is source code of the Java Program to display the Radio Buttons and Check Boxes selected. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Display Radio Buttons and Check Boxes selected*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Button_Checkbox implements ActionListener,ItemListener
6. {
7.     static JFrame frame;
8.     static JLabel text1,text2;
9.     static JCheckBox[] checkbox;
10.    static JRadioButton[] button;
11.    //Driver function
12.    public static void main(String args[])
13.    {
14.        //Create a frame
15.        frame=new JFrame("Buttons & Checkboxes");
16.        frame.setSize(600,600);
17.        frame.setLayout(null);
18.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19.        frame.getContentPane().setBackground(Color.white);
20.        //Create the text fields
21.        text1=new JLabel("");
22.        text1.setBounds(0,450,600,50);
23.        frame.add(text1);
24.        text2=new JLabel("");
25.        text2.setBounds(0,500,600,50);
26.        frame.add(text2);
27.        //Create an object
28.        Button_Checkbox obj=new Button_Checkbox();
29.        //Create 3 buttons
30.        button=new JRadioButton[3];
31.        for(int i=0;i<3;i++)
32.        {
33.            button[i]=new JRadioButton("Button "+(i+1));
34.            button[i].setBounds(200,i*80,100,50);
35.            frame.add(button[i]);
36.            button[i].addActionListener(obj);
37.        }
38.        //Create 3 Checkbox
39.        checkbox=new JCheckBox[3];
40.        for(int i=0;i<3;i++)
41.        {
42.            checkbox[i]=new JCheckBox("Checkbox "+(i+1));
43.            checkbox[i].setBounds(220,(240)+i*80,100,50);
44.            frame.add(checkbox[i]);
45.            checkbox[i].addItemListener(obj);
46.        }
47.        //Display frame
48.        frame.setVisible(true);
49.    }
50.    //To display button selected
51.    public void actionPerformed(ActionEvent e)
52.    {
53.        String s="";
54.        for(int i=0;i<3;i++)
55.        {
56.            if(button[i].isSelected())
57.                s=s+" "+button[i].getText();
58.        }
59.        text1.setText("Button(s) Selected : "+ " "+s);
60.    }
```

```
61. //To display the checkboxes checked
62. public void itemStateChanged(ItemEvent e)
63. {
64.     String s="";
65.     for(int i=0;i<3;i++)
66.     {
67.         if(checkbox[i].isSelected())
68.             s=s+" "+checkbox[i].getText();
69.     }
70.     text2.setText("Checkbox(s) Selected : "+s);
71. }
72. }
```

Program Explanation

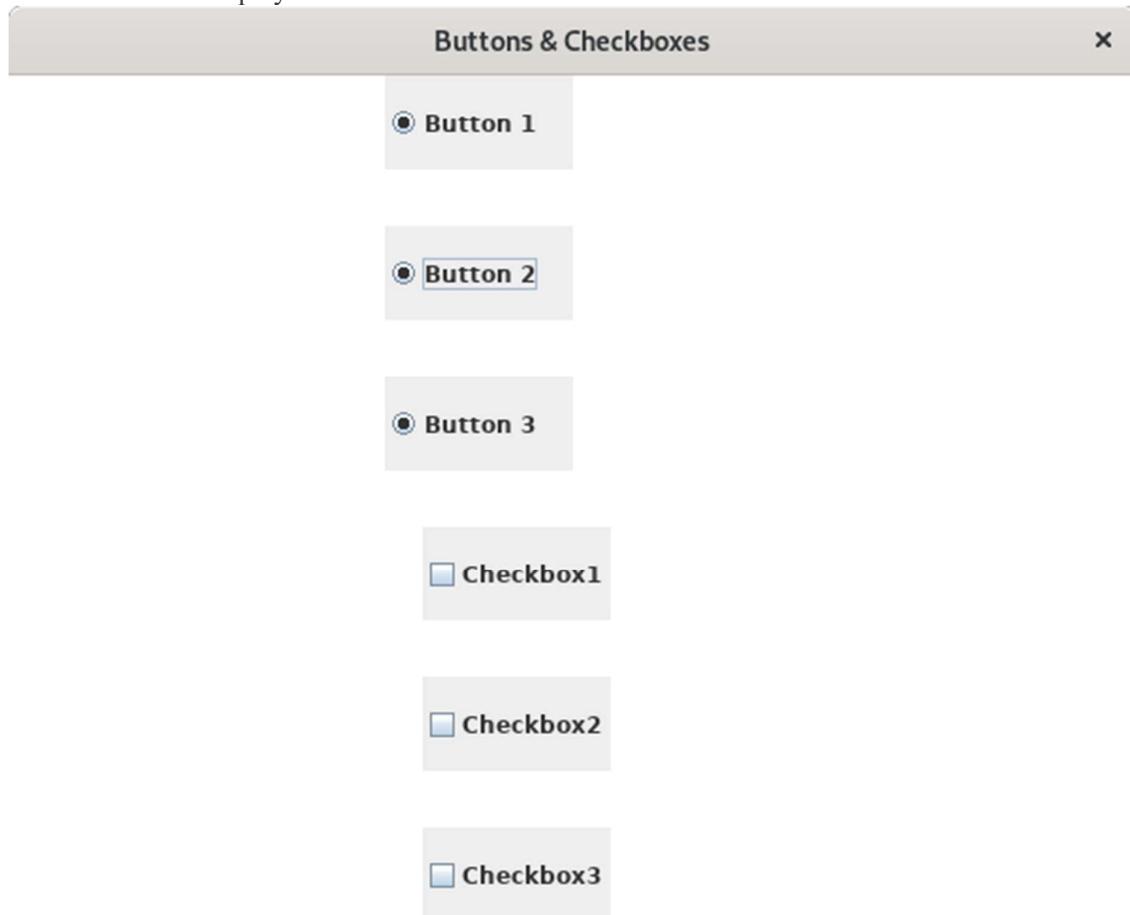
1. A frame is created with three Radio Buttons and Check Boxes.
2. Function **isSelected** is used to get the buttons/checkboxes selected.
3. Appropriate messages is displayed in the labels.

advertisement

Runtime Test Cases

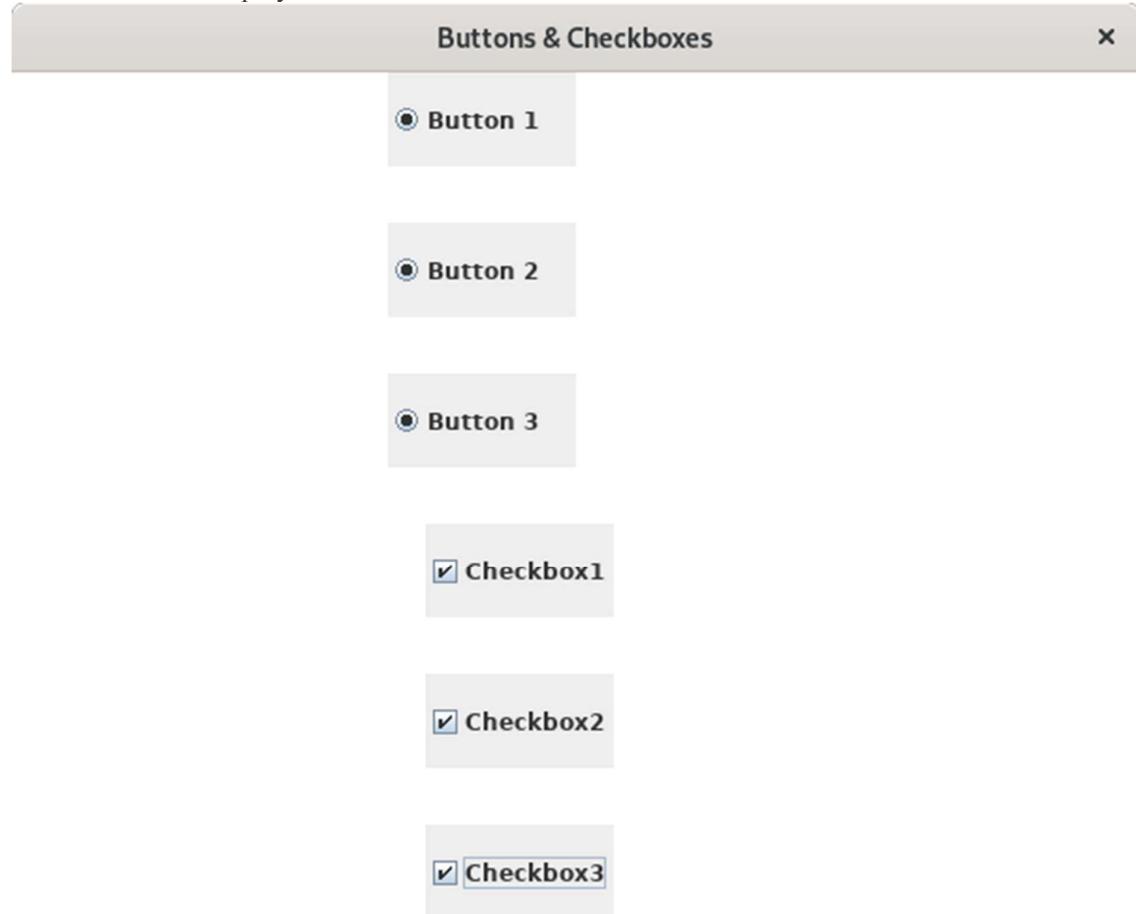
Here's the run time test cases to display the Radio Buttons and Check Boxes selected.

Test case 1 – To display the Radio Buttons Selected.



Button(s) Selected : Button 1 Button 2 Button 3

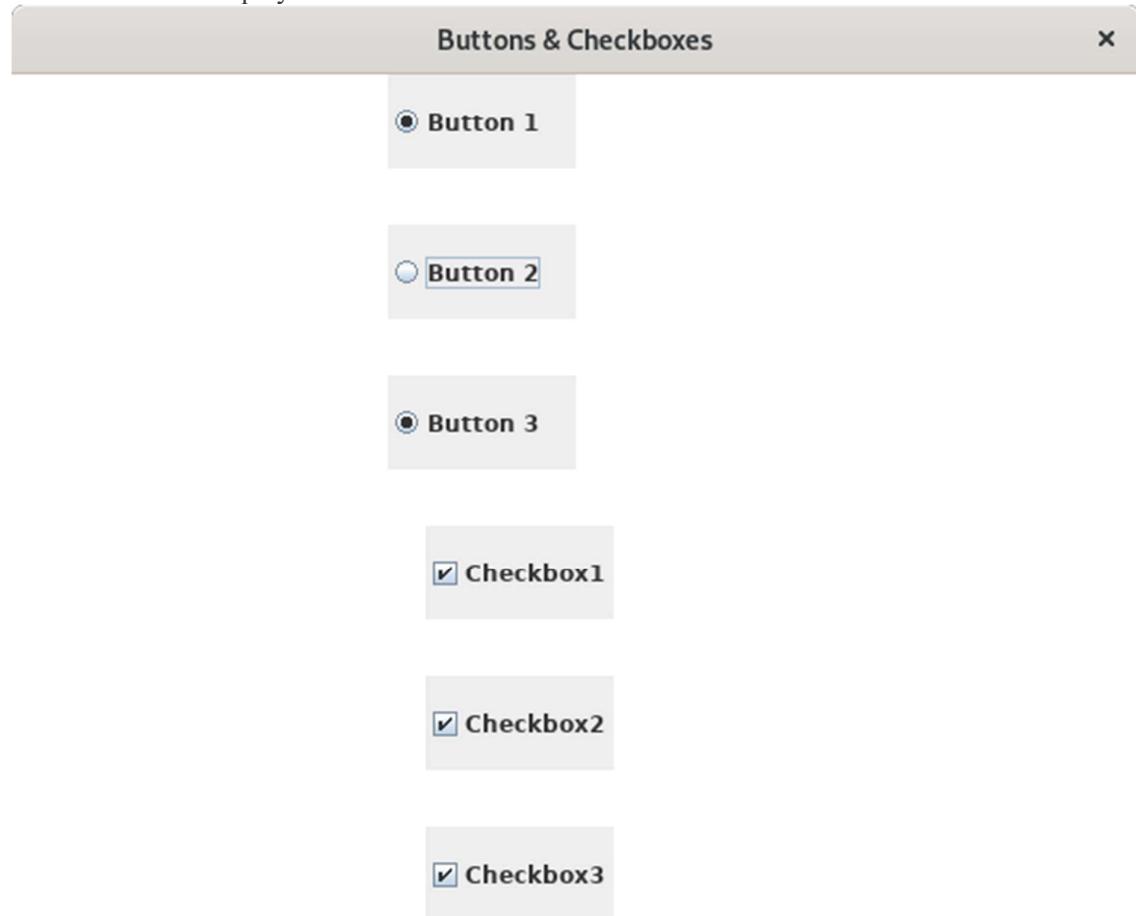
Test case 2 – To display the Check Boxes Selected.



Button(s) Selected : Button 1 Button 2 Button 3

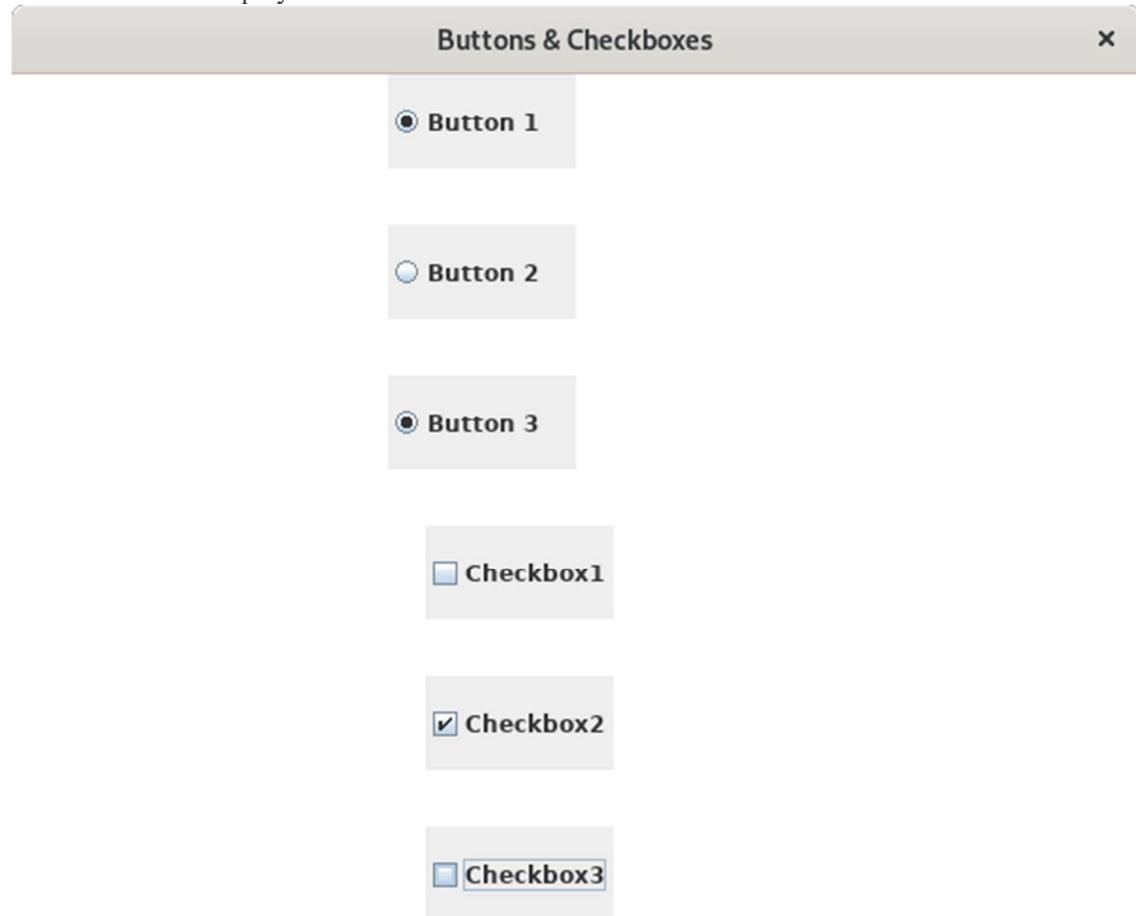
Checkbox(s) Selected : Checkbox1 Checkbox2 Checkbox3

Test case 3 – To display the Radio Buttons after De-Selected.



advertisement

Test case 4 – To display the Check Boxes after De-Selected.



Button(s) Selected : Button 1 Button 3

Checkbox(s) Selected : Checkbox2

150. Java Program that Changes the Look and Feel of the Component

[« Prev](#)

[Next »](#)

This is a Java Program that Changes the Look and Feel of the Component

Problem Description

We have to write a program in Java such that it allows the user to switch between three Look and Feel – Metal, Windows and Motif using radio buttons.

Expected Input and Output

To switch the Look and Feel of the Component, we have the following sets of input and output.

advertisement

1. Metal Look and Feel:

When the Metal Look and Feel is selected,
then it is expected that frame has the default Java API.

2. Windows Look and Feel:

When the Windows Look and Feel is selected,
then it is expected that frame has the Windows System API.

3. Motif Look and Feel:

advertisement

When the Motif Look and Feel is selected,
then it is expected that frame has the Motif/Linux API.

Problem Solution

1. Define a constructor of the class to create the components of the frame as required.
2. Change the Look and Feel of the component based on the option selected.
 - a) **Metal:** This is the default API of the Java framework.
 - b) **Windows:** This is the Windows System API.
 - c) **Motif:** This is the Linux System API for systems with GTK+ 2.2 or later.
3. Update the Look & Feel of the components of frame.

advertisement

Program/Source Code

Here is source code of the Java Program to change Look and Feel of Components. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Change Look and Feel*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. import javax.swing.SwingUtilities;
6. import javax.swing.UIManager;
7.
8. class Look_Feel implements ItemListener
9. {
10.    JFrame frame;
11.    JLabel label;
12.    JRadioButton radio[];
13.    UIManager.LookAndFeelInfo looks[] = UIManager.getInstalledLookAndFeels();
```

```

14. String UI_Names[]{"Metal","Windows","Motif"};
15. //Constructor to create the components
16. Look_Feel()
17. {
18.     //Create a frame
19.     frame=new JFrame("Change Look & Feel");
20.     frame.setSize(500,500);
21.     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22.     frame.setLayout(null);
23.     //Create a label
24.     label=new JLabel();
25.     label.setBounds(150,50,500,50);
26.     frame.add(label);
27.     //Create 3 buttons
28.     ButtonGroup group=new ButtonGroup();
29.     radio=new JRadioButton[3];
30.     for(int i=0;i<3;i++)
31.     {
32.         radio[i]=new JRadioButton(UI_Names[i]);
33.         radio[i].addItemListener(this);
34.         radio[i].setBounds(i*200,250,100,50);
35.         group.add(radio[i]);
36.         frame.add(radio[i]);
37.     }
38.     //Display the default UI
39.     label.setText("This is Metal Look & Feel");
40.     radio[0]. setSelected(true);
41.     //Display the frame
42.     frame.setVisible(true);
43. }
44. //Function to change the Look and Feel
45. public void changeLookAndFeel(int index)
46. {
47.     try
48.     {
49.         label.setText("This is "+UI_Names[index]+" Look & Feel");
50.         UIManager.setLookAndFeel(looks[index].getClassName());
51.         SwingUtilities.updateComponentTreeUI(frame);
52.     }
53.     catch (Exception e)
54.     {
55.         System.out.println("Error. UI Not Found in System");
56.     }
57. }
58. //Function to get the button selected
59. public void itemStateChanged(ItemEvent e)
60. {
61.     for(int i=0;i<3;i++)
62.     {
63.         if(radio[i].isSelected())
64.         {
65.             changeLookAndFeel(i);
66.             break;
67.         }
68.     }
69. }
70. //Driver function
71. public static void main(String args[])
72. {
73.     Look_Feel obj=new Look_Feel();
74. }
75.

```

Program Explanation

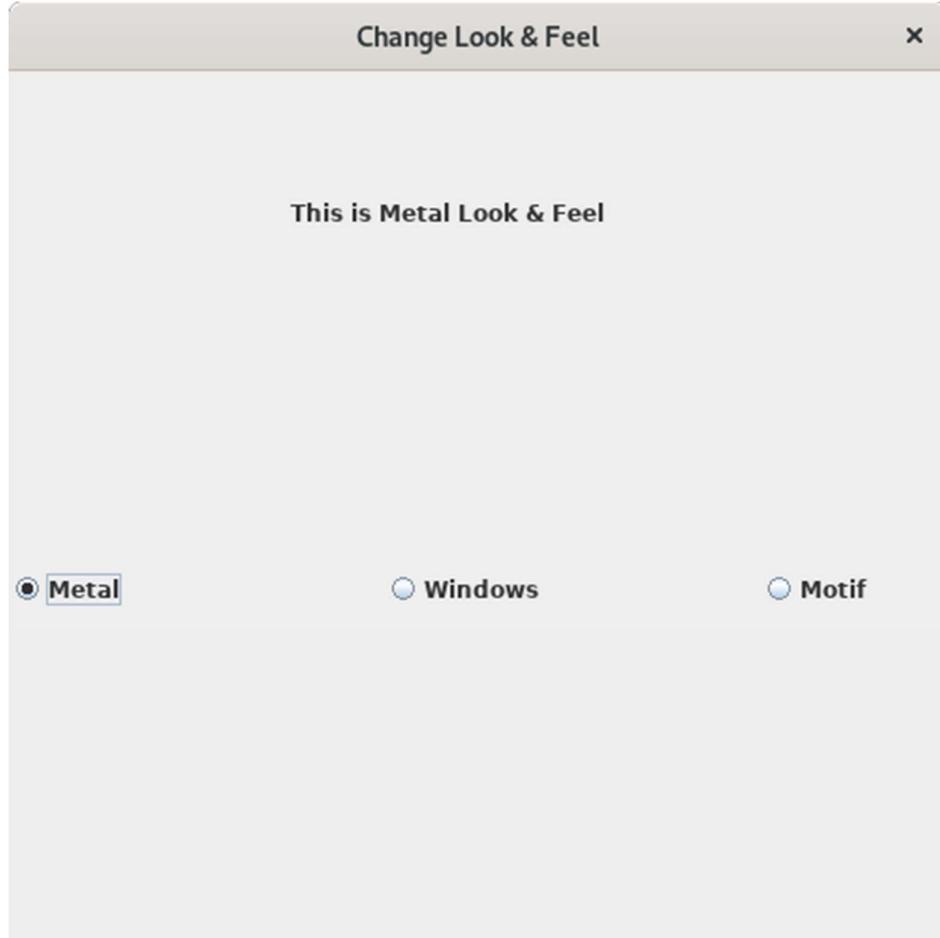
1. Get the list of installed Look and Feels using function `getInstalledLookAndFeels` of the `UIManager` class.
2. Create the components of the frame as required.
3. Associate **ItemListener** with the radio buttons.
4. When a radio button is selected, get the index of the look and feel selected and call function `changeLookAndFeel` to change the look and feel to the selected.
5. Change the look and feel using `updateComponentTreeUI` of the `SwingUtilities` class.

Runtime Test Cases

Here's the run time test cases to change the Look and Feel of the Component.

advertisement

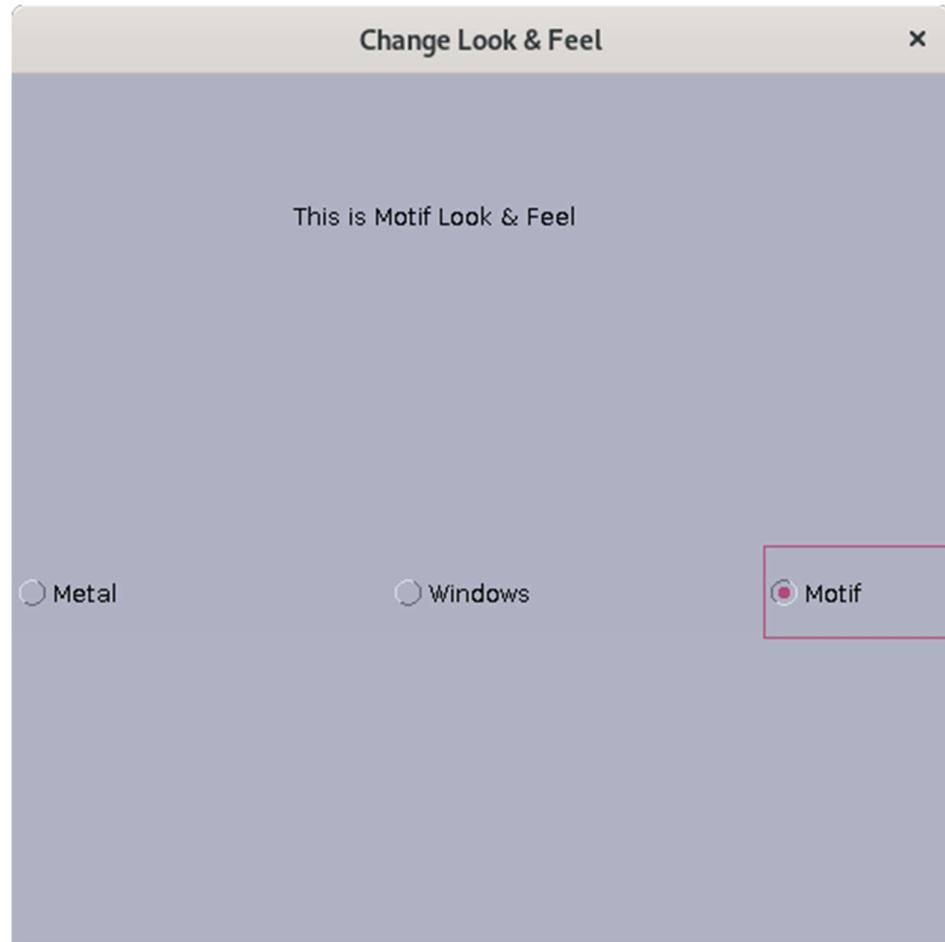
Test case 1 – For the Metal Look and Feel



Test case 2 – For the Windows Look and Feel



Test case 3 – For the Motif Look and Feel



151. Java Program to Create a Split Pane with Two Parts to Display Button and Display Text inside Text Area when Button is Clicked

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Split Pane with Two Parts to Display Button and Display Text inside Text Area when Button is Clicked

Problem Description

We have to write a program in Java such that it creates a vertically split frame where the left part consists of few push buttons and the right part consists of a text area. When a push button is clicked, the button is displayed in the text area.

Expected Input and Output

To create a Split Pane, we have the following sets of input and output.

advertisement

1. To view the frame:

At the execution of the program,
it is expected that a vertically split frame appears.
left part frame displays 5 push buttons and right part displays text area

2. To view text in the text area:

advertisement

Suppose there are 5 push buttons in left part of frame
(Button 1, Button 2, Button 3, Button 4, Button 5)
When any push button is clicked for example (Button 1 is clicked),
then it is expected that the text area shows "Button clicked: Button 1".

Problem Solution

1. Create a frame and two panels for the left part and the right part.
2. Add few buttons to the left part, and a text area to the right part.
3. Create a Split Pane using **JSplitPane** and add the left and right components to it.
4. Add the Split Pane to the frame and display the frame.

Program/Source Code

Here is source code of the Java Program to create a Split Pane. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to Create a Split Pane*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Split_Pane implements ActionListener
6. {
7.     static JTextArea text;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame = new JFrame("Tabbed Pane");
13.        frame.setSize(750,250);
```

```

14. frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15. //Create an object of the class
16. Split_Pane obj = new Split_Pane();
17. //Create Panel 1 for Left Part
18. JPanel panel1 = new JPanel(false);
19. JButton button[] = new JButton[5];
20. for(int i=0;i<5;i++)
21. {
22.     button[i] = new JButton("Button "+(i+1));
23.     button[i].addActionListener(obj);
24.     panel1.add(button[i]);
25. }
26. //Create Panel 2 for Right Part
27. JPanel panel2 = new JPanel(false);
28. text = new JTextArea(5,20);
29. panel2.add(text);
30. //Create a split pane
31. JSplitPane pane = new JSplitPane(SwingConstants.VERTICAL);
32. pane.setLeftComponent(panel1);
33. pane.setRightComponent(panel2);
34. //Add split pane to frame
35. frame.add(pane);
36. //Display the frame
37. frame.setVisible(true);
38. }
39. //Function to display the button clicked
40. public void actionPerformed(ActionEvent e)
41. {
42.     String button = e.getActionCommand();
43.     text.setText("Button Clicked : "+button);
44. }
45.

```

Program Explanation

1. Panel 1 is the left component of the split pane, and it consists of push buttons.
2. Panel 2 is the right component of the split pane, and it consists of a text area.
3. A vertical split pane is created using **JSplitPane**. To set the spit orientation to vertical **SwingConstants.VERTICAL** is used.
4. The left and right components are set using **setLeftComponent** and **setRightComponent** respectively.
5. The split pane is added to the frame and the frame is displayed.
6. When a button is clicked, the **actionPerformed** function is called by the **ActionListener** and it displays the button clicked in the text area.

Runtime Test Cases

Here's the run time test cases to create a Split Pane.

advertisement

Test case 1 – To View the Split Pane



Test case 2 – To View the Button 1 Clicked



Test case 3 – To View the Button 4 Clicked



152. Java Program to Create a Combo Box to Select One Option and Display the Label of Option Selected

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Combo Box to Select One Option and Display the Label of Option Selected

Problem Description

We have to write a program in Java such that it creates a Combo Box containing name of some countries, and the option selected by user is displayed in the frame.

Expected Input and Output

For displaying label of option selected from a Combo Box, we can have the following set of input and output.

advertisement

To display the label:

When any option on the Combo Box is selected by the user, it is expected that the label of the item selected is displayed.

Problem Solution

1. Create a combo box using the **JComboBox** class.
2. Add the list of countries to it.
3. Associate **ItemListener** with the combo box.
4. When any item is selected, display the label of item in the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to view the label of item selected from a Combo Box. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to view label of item selected from a Combo Box*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Combo_Box implements ItemListener
6. {
7.     static JLabel text;
8.     static JComboBox<String> box;
9.     //Driver function
10.    public static void main(String args[])
11.    {
12.        //Create a frame
13.        JFrame frame = new JFrame("Combo Box");
14.        frame.setSize(500,500);
15.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.        frame.setLayout(null);
17.        //Create an object
18.        Combo_Box obj = new Combo_Box();
19.        //Create a label
20.        text = new JLabel("Select an Item");
21.        text.setBounds(175,50,200,50);
22.        frame.add(text);
23.        //Create an array with countries name
```

```

24. String countries[]={"India","Japan","Nepal","China","Sri Lanka"};
25. //Create a Combo Box
26. box = new JComboBox<String>(countries);
27. box.setBounds(200,200,100,50);
28. box.addItemListener(obj);
29. frame.add(box);
30. //Display the frame
31. frame.setVisible(true);
32. }
33. //Function to view the label of item selected
34. public void itemStateChanged(ItemEvent e)
35. {
36. text.setText("Item Selected : "+box.getSelectedItem());
37. }
38.

```

Program Explanation

1. Create a Combo Box using **JComboBox**, with **String** data type.

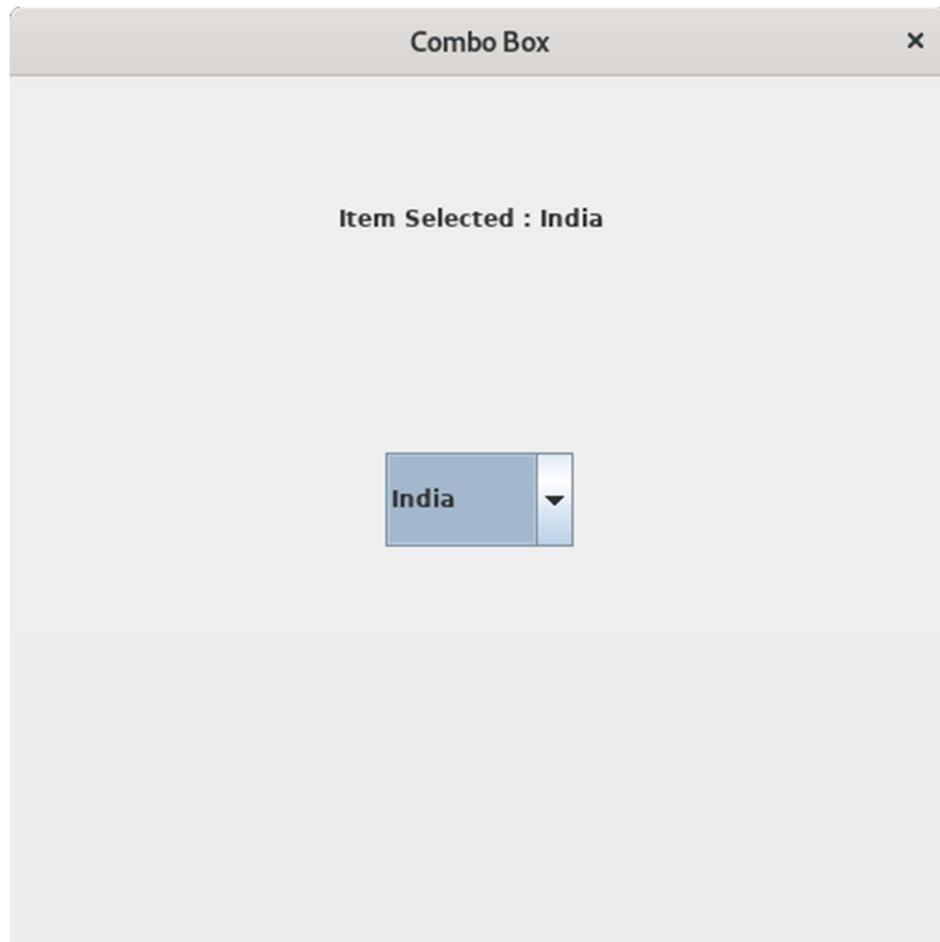
2. To know which item was selected, using **getSelectedItem**.

advertisement

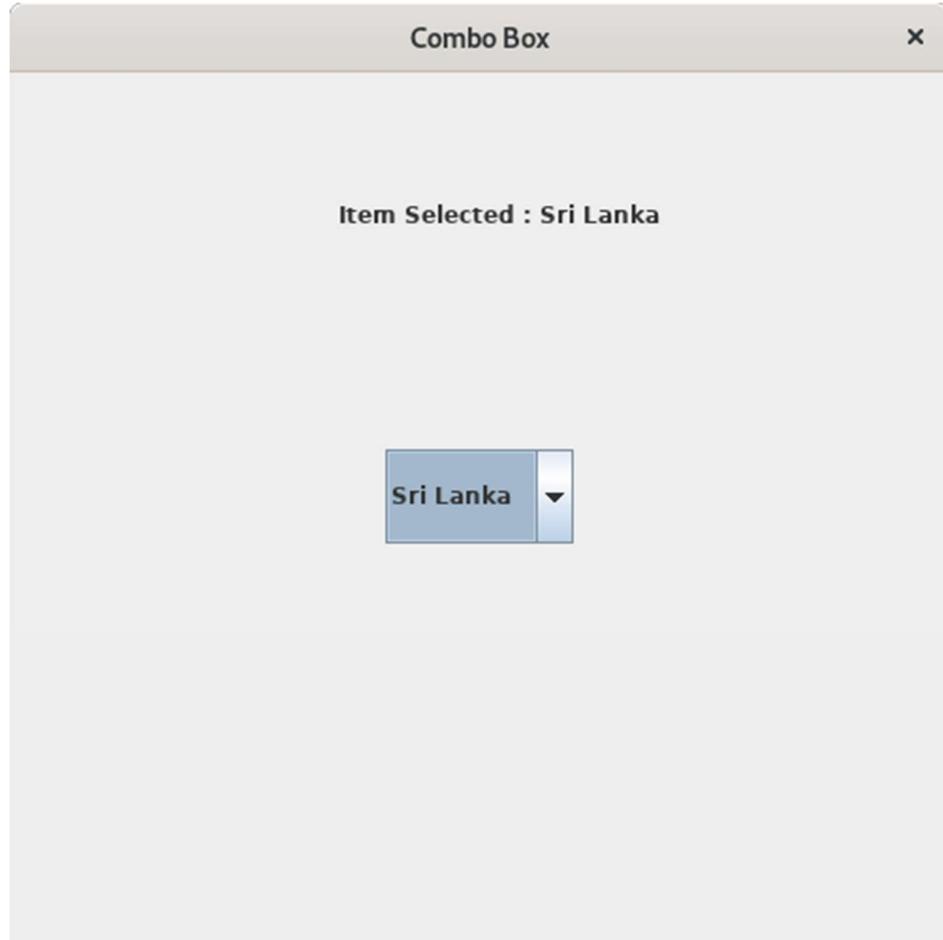
Runtime Test Cases

Here's the run time test cases for viewing the label of item selected from a combo box.

Test case 1 – To view the label of item selected.

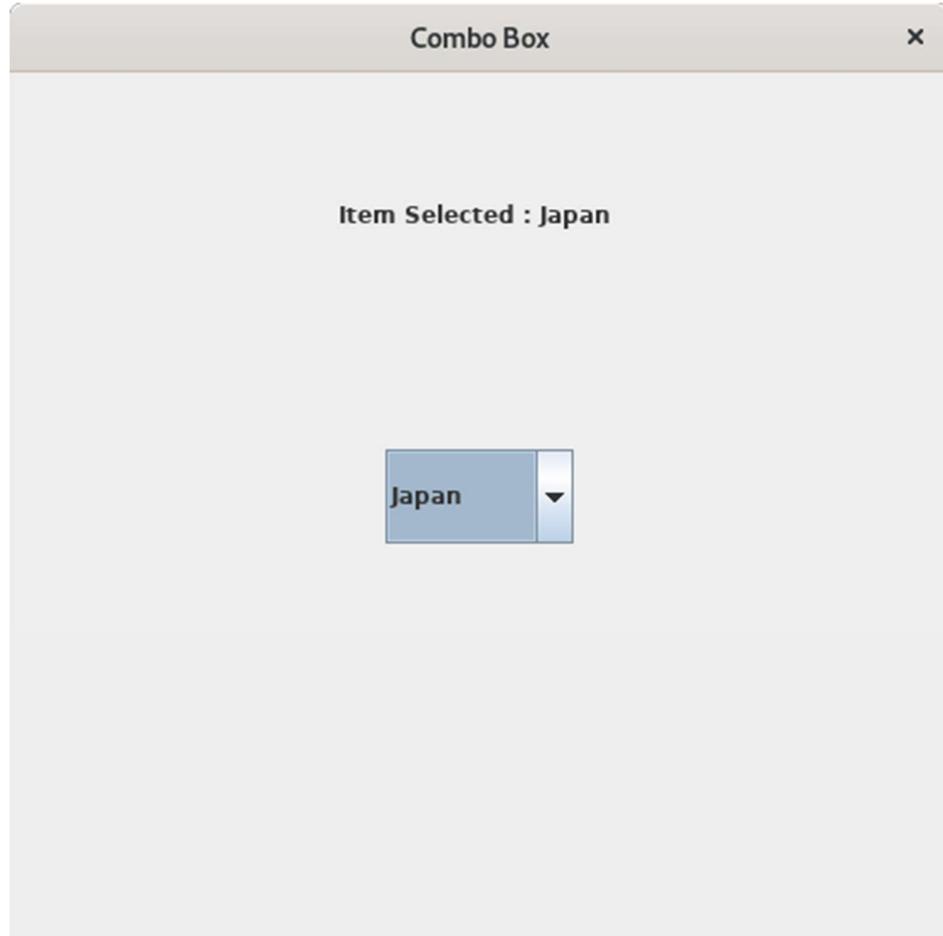


Test case 2 – To view the label of item selected.



advertisement

Test case 3 – To view the label of item selected.



153. Java Program to Create a Menu with Several Menu Items

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Menu with Several Menu Items

Problem Description

We have to write a program in Java such that it creates a menu bar with menu items and the the label of the menu item is displayed in the frame.

Expected Input and Output

For creating a menu with several items, we can have the following sets of input and output.

advertisement

1.To View the Menu:

When the program is being executed,
it is expected that the frame consists of the menu.

2.To Select a Menu Item:

advertisement

When any menu item is selected from the menu bar,
it is expected that the label of the menu item selected is displayed.

Problem Solution

1. Create a menu and add menu items to it.
2. Add **ActionListener** to all menu items.
3. Create a menu bar and add the menu to it.
4. Create a label and display the menu item selected.

Program/Source Code

Here is source code of the Java Program to view the label of item selected from a Menu. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to Create a Menu and Display the Menu Item Selected*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Menu implements ActionListener
6. {
7.     static JLabel text;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame = new JFrame("Menu");
13.        frame.setSize(500,500);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        frame.setLayout(new FlowLayout());
16.        //Create an object
17.        Menu obj = new Menu();
18.        //Create a Menu
19.        JMenu menu = new JMenu("Select Here");
20.        //Create Menu Items
```

```

21. JMenuItem item[] = new JMenuItem[5];
22. for(int i=0;i<5;i++)
23. {
24.     item[i]=new JMenuItem("Item "+(i+1));
25.     item[i].addActionListener(obj);
26.     menu.add(item[i]);
27. }
28. //Create a menu bar
29. JMenuBar mb=new JMenuBar();
30. mb.add(menu);
31. frame.setJMenuBar(mb);
32. //Create the label
33. text = new JLabel();
34. frame.add(text);
35. //Display the frame
36. frame.setVisible(true);
37. }
38. //Function to display the menu item selected
39. public void actionPerformed(ActionEvent e)
40. {
41.     text.setText("Menu Item Selected : "+e.getActionCommand());
42. }
43.

```

Program Explanation

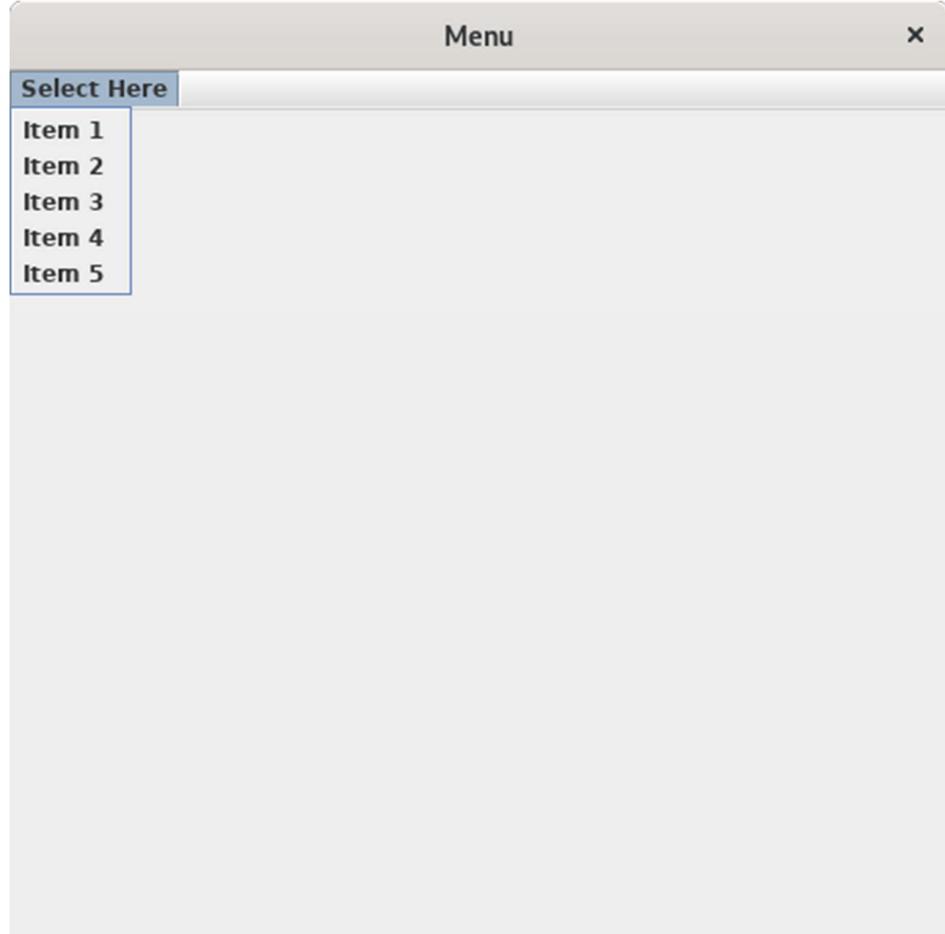
1. Create a menu using **JMenu** class.
2. To create menu items use the **JMenuItem** class.
3. Create a menu bar using **JMenuBar** and add the menu to it.
4. Set the menu to the frame using **setJMenuBar** function.
5. Display the label of the menu item selected.

Runtime Test Cases

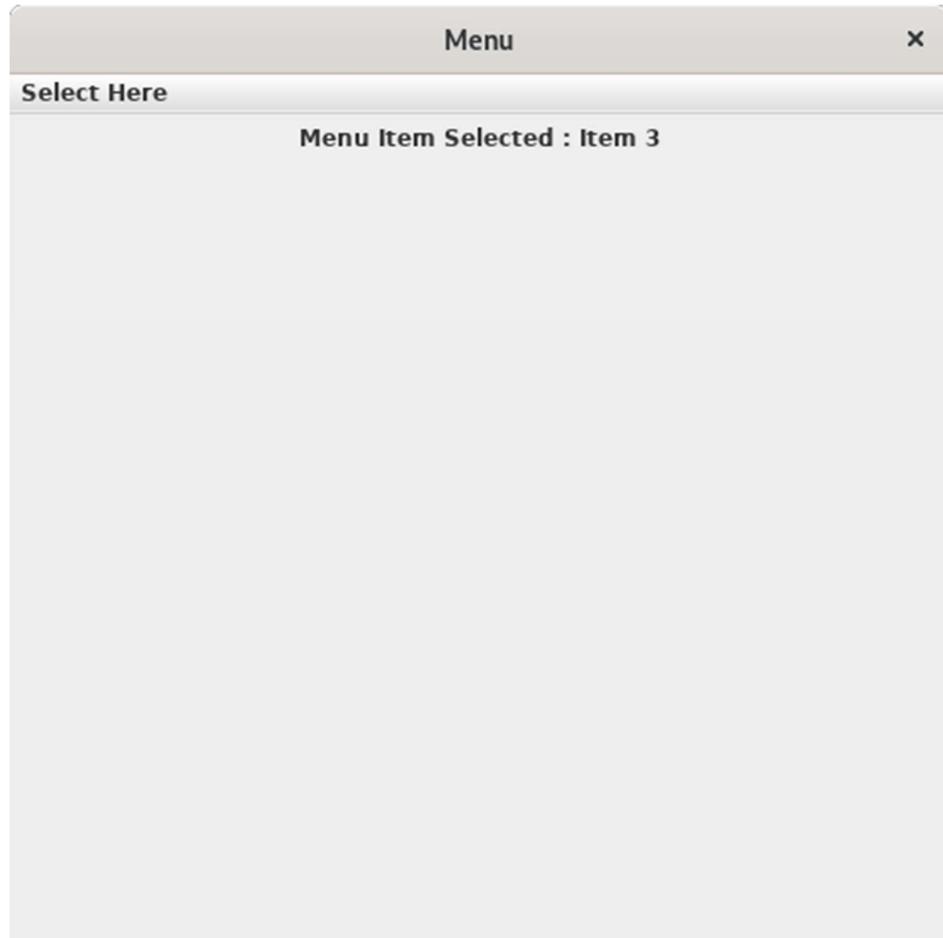
Here's the run time test cases for selecting an item from a menu.

advertisement

Test case 1 – To View the Menu.



Test case 2 – To View the Menu Item Selected.



Test case 3 – To View the Menu Item Selected.



154. Java Program to Create a Menu and Handle the File Open Event for the User

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Menu and Handle the File Open Event for the User

Problem Description

We have to write a program in Java such that it creates a frame with a menu item to handle the File Open event. By File Open event it means that the user shall be able to browse the system and select a file. The file path of the selected file will be displayed in the frame.

Expected Input and Output

For handling the File Open event, we can have the following set of input and output.

advertisement

To display the file selected:

When any file item is selected from the file browser window, it is expected that the complete file path is displayed in the frame.

Problem Solution

1. Create a Menu with the **Open** menu item.
2. When the user clicks on **Open**, a file browser window is opened.
3. The path of file selected is displayed in the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to handle the file open event. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to handle the File Open Event*/
2. import javax.swing.*;
3. import javax.swing.filechooser.*;
4. import java.awt.*;
5. import java.awt.event.*;
6. class Open_File implements ActionListener
7. {
8.     static JFrame frame;
9.     static JLabel label;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.        //Create a frame
14.        frame = new JFrame("Open a File");
15.        frame.setSize(500,500);
16.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17.        frame.setLayout(new FlowLayout());
18.        //Create a label
19.        label = new JLabel();
20.        frame.add(label);
21.        //Create an object
22.        Open_File obj = new Open_File();
23.        //Create a Menu
24.        JMenu menu = new JMenu("File");
25.        //Create a Menu Item
26.        JMenuItem open = new JMenuItem("Open");
```

```

27. open.addActionListener(obj);
28. menu.add(open);
29. //Create a menu bar
30. JMenuBar mb=new JMenuBar();
31. mb.add(menu);
32. frame.setJMenuBar(mb);
33. //Display the frame
34. frame.setVisible(true);
35. }
36. //Function to create a file chooser and display the path
37. public void actionPerformed(ActionEvent e)
38. {
39. //Create a file chooser
40. JFileChooser file = new JFileChooser();
41. file.showOpenDialog(null);
42. label.setText("File : "+file.getSelectedFile());
43. }
44.

```

Program Explanation

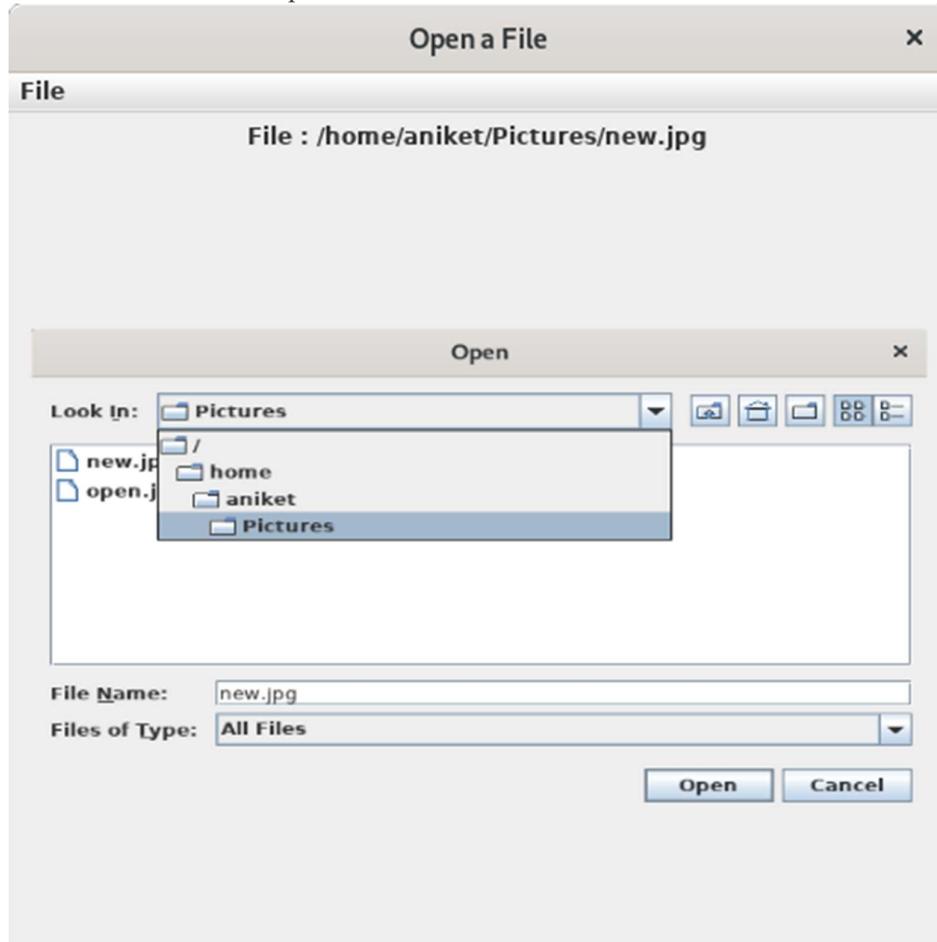
1. To create a file chooser use the **JFileChooser** class.
2. To view the file chooser, use **showOpenDialog** method.
3. Display the complete path of file.

advertisement

Runtime Test Cases

Here's the run time test case to handle the file open event.

Test case – To view the path of the file.



155. Java Program to Create a Toolbar with 3 Push Buttons with Image and Display the Selected Button in the Label

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Toolbar with 3 Push Buttons with Image and Display the Selected Button in the Label

Problem Description

We have to write a program in Java such that it creates a toolbar with 3 buttons – New, Open and Print with images. When any button on the toolbar is clicked, the button is displayed in the label.

Expected Input and Output

Consider the following icons are used for the push buttons :-

a) **For New Button :** This image is saved as “new.jpg”

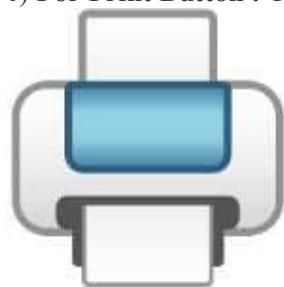


advertisement

b) **For Open Button :** This image is saved as “open.jpg”



c) **For Print Button :** This image is saved as “print.jpg”



For creating a toolbar with push buttons having an image, we can have the following different sets of input and output.

1. To Display the Toolbar:

advertisement

When the program is being executed, it is expected that a toolbar is created with 3 push buttons having images. push buttons - New, Open and Print with images

2. To Display the Button clicked: Open

When open button on the toolbar is clicked,
it is expected that a open button is displayed in a label.

3. To Display the Button clicked: New

advertisement

When new button on the toolbar is clicked,
it is expected that a new button is displayed in a label.

4. To Display the Button clicked: Print

When print button on the toolbar is clicked,
it is expected that a print button is displayed in a label.

Problem Solution

1. Create 3 image icons and 3 push buttons. Add image icons to the buttons.
2. Create a toolbar and add the 3 push buttons to it.
3. Add the toolbar to the image.
4. When any button is clicked, create an object of the **JButton** class to get the button.
5. Obtain the image icon of the button clicked and compare it with the initially set icons.
6. Display the appropriate message.

Program/Source Code

Here is source code of the Java Program to create a toolbar with push buttons having images. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /* Java Program to create a toolbar with push buttons having images*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. import java.awt.image.*;
6. class Toolbar implements ActionListener
7. {
8.     static JLabel text;
9.     static ImageIcon icon_open,icon_new,icon_print;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.        //Create a frame
14.        JFrame frame = new JFrame("Toolbar");
15.        frame.setSize(600,400);
16.        frame.getContentPane().setBackground(Color.white);
17.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18.        frame.setLayout(new FlowLayout());
19.        //Create an object
20.        Toolbar obj =new Toolbar();
21.        //Create open button
22.        icon_open = new ImageIcon("open.jpg");
23.        JButton b_open = new JButton(icon_open);
24.        b_open.addActionListener(obj);
25.        //Create new button
26.        icon_new = new ImageIcon("new.jpg");
27.        JButton b_new = new JButton(icon_new);
28.        b_new.addActionListener(obj);
29.        //Create print button
30.        icon_print = new ImageIcon("print.jpg");
31.        JButton b_print = new JButton(icon_print);
32.        b_print.addActionListener(obj);
33.        //Create a toolbar
34.        JToolBar bar = new JToolBar();
35.        bar.add(b_open);
36.        bar.add(b_new);
```

```

37. bar.add(b_print);
38. frame.add(bar);
39. //Create a label
40. text = new JLabel();
41. frame.add(text);
42. //Display the frame
43. frame.setVisible(true);
44. }
45. //Function to view the button clicked
46. public void actionPerformed(ActionEvent e)
47. {
48.     //Create a button of the action event source
49.     JButton button = (JButton)e.getSource();
50.     //Get the icon of button
51.     String icon = button.getIcon().toString();
52.     //Display the button clicked
53.     if(icon.equals(icon_open.toString()))
54.         text.setText("Button Clicked : Open");
55.     else if(icon.equals(icon_new.toString()))
56.         text.setText("Button Clicked : New");
57.     else
58.         text.setText("Button Clicked : Print");
59. }
60.

```

Program Explanation

1. The image icons **icon_open**, **icon_new** and **icon_print** are the icons for the **Open**, **New** and **Print** respectively.
2. Create a toolbar using **JToolBar** class.
3. To create an button from the action event source, explicitly convert the event source to a button using **JButton**.
4. Get the string of the image icon of button using **getIcon().toString()**.

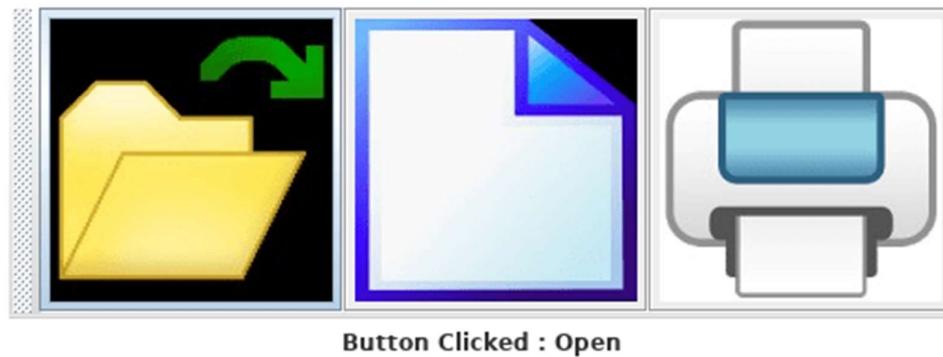
Runtime Test Cases

Here's the run time test case to create a toolbar with push buttons having images.

Test case 1 – To View the Toolbar with Push Buttons.



Test case 2 – To View the Button Clicked – Open.



advertisement

Test case 3 – To View the Button Clicked – New.



Test case 4 – To View the Button Clicked – Print.



156. Java Program to Show the Functioning of a Toggle Button

[« Prev](#)

[Next »](#)

This is a Java Program to Show the Functioning of a Toggle Button

Problem Description

We have to write a program in Java such that it shows the functioning of a toggle button.

Toggle Button : It is a two-state button that allows the user to switch the button either on or off, that is the status of the button can be switched between selected and deselected respectively.

Expected Input and Output

For showing the function of Toggle Button, we can have the following different sets of input and output.

advertisement

1. To Display the Toggle Button:

When the program is being executed,
it is expected that a toggle button is created and displayed.

2. To Display the Selected Status of Toggle Button:

advertisement

When the toggle button is selected,
the expected output is "Status of Toggle Button : SELECTED"

3. To Display the Deselected Status of Toggle Button:

When the toggle button is deselected,
the expected output is "Status of Toggle Button : DESELECTED"

Problem Solution

1. Create a frame and a label.
2. Create a Toggle Button and add **ItemListener** to it.
3. When any action is performed on the toggle button, display the appropriate message.

advertisement

Program/Source Code

Here is source code of the Java Program to show the functioning of Toggle Button. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to show the functioning to Toggle Button*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. import javax.swing.JToggleButton;
6. class Toggle implements ItemListener
7. {
8.     static JLabel text;
9.     //Driver function
10.    public static void main(String args[])
11.    {
12.        //Create a frame
13.        JFrame frame = new JFrame("Toggle Button");
14.        frame.setSize(500,500);
15.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.        frame.getContentPane().setBackground(Color.white);
17.        frame.setLayout(null);
18.        //Create an object
```

```
19. Toggle obj = new Toggle();
20. //Create a Toggle Button
21. JToggleButton button = new JToggleButton("Toggle Button");
22. button.setBounds(150,100,200,100);
23. button.addItemListener(obj);
24. frame.add(button);
25. //Create a label
26. text = new JLabel();
27. text.setBounds(150,250,300,100);
28. text.setText("Click on the Toggle Button");
29. frame.add(text);
30. //Display the frame
31. frame.setVisible(true);
32. }
33. //Function to display the status of toggle button
34. public void itemStateChanged(ItemEvent e)
35. {
36. if(e.getStateChange()==ItemEvent.SELECTED)
37. text.setText("Status of Toggle Button : SELECTED");
38. else
39. text.setText("Status of Toggle Button : DESELECTED");
40. }
41.}
```

Program Explanation

1. To create a toggle button use **JToggleButton** class.
2. Add **ItemListener** to the toggle button.
3. Use method **getStateChange** to get the state change of toggle button.
advertisement

Runtime Test Cases

Here's the run time test case to show the functioning of toggle button.

Test case 1 – To View the Toggle Button.



Click on the Toggle Button

Test case 2 – To View the Selected status of Toggle Button.



Status of Toggle Button : SELECTED

Test case 3 – To View the Deselected status of Toggle Button.



Status of Toggle Button : DESELECTED

157. Java Program to Progress a Progress Bar by 5 Units on Every Click of Push Button

[« Prev](#)

[Next »](#)

This is a Java Program to Progress a Progress Bar by 5 Units on Every Click of Push Button

Problem Description

We have to write a program in Java such that it creates a Progress Bar and a Push Button. Every time the push button is clicked the progress bar is progressed by 5 units.

Expected Input and Output

For progressing the progress bar, we can have the following different sets of input and output.

advertisement

1. To Display the Progress Bar:

When the program is being executed, it is expected that a progress bar is created with initial value zero.

2. To Progress the Progress Bar:

advertisement

Every time the push button is clicked, it is expected that the progress bar progresses by 5 units.

Problem Solution

1. Create a frame containing a progress bar and push button.
2. Set the initial value of the progress bar to zero.
3. Every time the push button is clicked, increase the progress bar by 5 units.

Program/Source Code

Here is source code of the Java Program to progress the progress bar. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to progress the progress bar*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Progress_Bar implements ActionListener
6. {
7.     static JProgressBar bar;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame = new JFrame("Progress Bar");
13.        frame.setSize(500,500);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        frame.getContentPane().setBackground(Color.white);
16.        frame.setLayout(null);
17.        //Create an object
18.        Progress_Bar obj = new Progress_Bar();
19.        //Create a Progress Bar
20.        bar = new JProgressBar();
```

```

21. bar.setValue(0);
22. bar.setStringPainted(true);
23. bar.setBounds(150,100,200,50);
24. frame.add(bar);
25. //Create a button
26. JButton increase = new JButton("Increase");
27. increase.setBounds(200,250,100,50);
28. increase.addActionListener(obj);
29. frame.add(increase);
30. //Display the frame
31. frame.setVisible(true);
32. }
33. //Function to progress the progress bar
34. public void actionPerformed(ActionEvent e)
35. {
36.     bar.setValue(bar.getValue() + 5);
37. }
38.

```

Program Explanation

1. Create a progress bar using **JProgressBar** class.
2. To set the value of the progress bar use **setValue** function.
3. To get the present value of progress bar use **getValue** function.
4. To display the value of the progress bar within the bar use **setStringPainted(true)**.

Runtime Test Cases

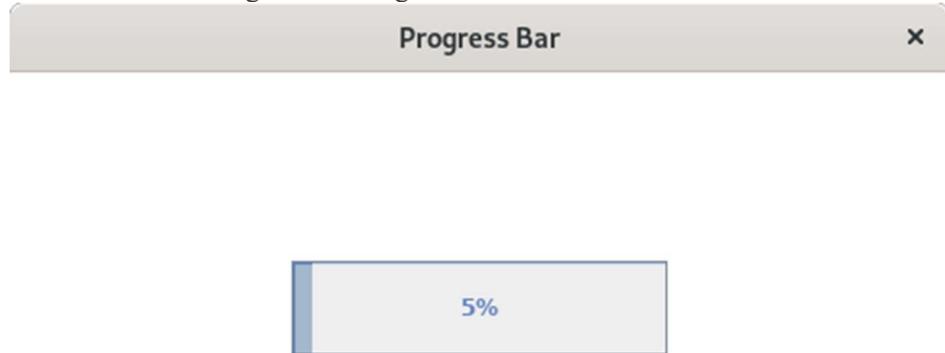
Here's the run time test case to show the progress the progress bar.

advertisement

Test case 1 – To View the Progress Bar.



Test case 2 – To Progress the Progress Bar.



158. Java Program to Create a Group of Buttons and Arrange them in Container using Flow Layout Manager

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Group of Right Justified Buttons and Arrange them in Container using Flow Layout Manager

Problem Description

We have to write a program in Java such that it creates a frame with few buttons which are right justified using the Flow Layout Manager.

Expected Input and Output

For creating frame with right justified buttons, we have the following set of input and output.

advertisement

1. When the frame is created :

On the execution of the program,
it is expected that a frame appears with a few right justified buttons.
Buttons - Button 1, Button 2, Button 3

Problem Solution

1. Create a frame and few buttons.
2. Set the layout of frame as **FlowLayout** with alignment of components as **RIGHT**
3. Display the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to create right justified buttons. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to create right justified buttons and
2. arrange in container using Flow Layout Manager*/
3. import javax.swing.*;
4. import java.awt.*;
5. import java.awt.FlowLayout;
6. class Push_Buttons
7. {
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        JFrame frame = new JFrame("Buttons");
13.        frame.setSize(500,300);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        //Set the layout of frame as right aligned flow layout
16.        frame.setLayout(new FlowLayout(FlowLayout.RIGHT));
17.        //Create 3 push buttons
18.        JButton[] button = new JButton[3];
19.        for(int i=0;i<3;i++)
20.        {
21.            button[i]=new JButton("Button "+(i+1));
22.            frame.add(button[i]);
```

```
23. }  
24. //Display the frame  
25. frame.setVisible(true);  
26. }  
27.}
```

Program Explanation

1. Set the layout of frame as **FlowLayout**.
2. Set the alignment of the layout as **FlowLayout.RIGHT**. This ensures that all the components are right aligned.

advertisement

Runtime Test Cases

Here's the run time test case for creating a frame with right justified buttons.

Test case 1 – To view the frame with buttons.



159. Java Program to Perform Menu Driven Arithmetic Operations Using Switch Case

[« Prev](#)

[Next »](#)

This is a Java Program to Perform Menu Driven Arithmetic Operations Using Switch Case

Problem Description

We have to write a program in Java such that it creates two input fields for two numbers, and contains four buttons for the operations – Addition, Subtraction, Multiplication and Division. When the user clicks on any button after entering the value in the two input field, the respective operation is performed and the value is displayed in an output text field.

Expected Input and Output

For performing arithmetic operations, we have the following different sets of input and output.

advertisement

1. When Addition is Selected :

If the user selects the Add option,
it is expected that the sum of the numbers is displayed.

For example, if we are entering the first number 10.5 and the second number is 20.
then addition of 2 numbers is $10.5 + 20 = 30.5$

2. When Subtraction is Selected :

If the user selects the Subtract option,
it is expected that the difference of the numbers is displayed.

For example, if we are entering first number 10.5 and the second number is 20.
then Subtraction of 2 numbers is $10.5 - 20 = -9.5$

3. When Multiplication is Selected :

advertisement

If the user selects the Multiply option,
it is expected that the product of the numbers is displayed.

For example, if we are entering the first number 10.5 and the second number is 20.
then product of 2 numbers is $10.5 * 20 = 210$

4. When Division is Selected :

If the user selects the Divide option,
it is expected that the quotient of the numbers when divided is displayed.
For example, if we are entering the first number 10.5 and the second number is 20.
then division of 2 numbers is $10.5/20 = 0.525$

Problem Solution

1. Create a frame with two input fields for the two numbers.
2. Create four buttons – **Add, Subtract, Multiply and Divide**.
3. When any button is clicked, perform the operation using switch case.
4. Display the result in an output text field.

advertisement

Program/Source Code

Here is source code of the Java Program to perform arithmetic operations. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Perform Arithmetic Operations Using Switch Case*/
2. import javax.swing.*;
```

```
3. import java.awt.*;
4. import java.awt.event.*;
5. class Arithmetic_Operations implements ActionListener
6. {
7.     static JFrame frame;
8.     static JTextField text1,text2;
9.     static JTextField out;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.        //Create a frame
14.        frame = new JFrame("Aritmetic Operations");
15.        frame.setSize(500,500);
16.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17.        frame.getContentPane().setBackground(Color.white);
18.        frame.setLayout(null);
19.        //Create an object
20.        Arithmetic_Operations obj = new Arithmetic_Operations();
21.        //Input field for first number
22.        JLabel label1 = new JLabel("First Number : ");
23.        label1.setBounds(100,50,150,40);
24.        frame.add(label1);
25.        text1 = new JTextField(5);
26.        text1.setBounds(250,50,100,40);
27.        frame.add(text1);
28.        //Input field for second number
29.        JLabel label2 = new JLabel("Second Number : ");
30.        label2.setBounds(100,150,150,40);
31.        frame.add(label2);
32.        text2 = new JTextField(5);
33.        text2.setBounds(250,150,100,40);
34.        frame.add(text2);
35.        //Create four buttons
36.        JButton add = new JButton("Add");
37.        add.setBounds(30,250,100,50);
38.        JButton sub = new JButton("Subtract");
39.        sub.setBounds(140,250,100,50);
40.        JButton mul = new JButton("Multiply");
41.        mul.setBounds(250,250,100,50);
42.        JButton div = new JButton("Divide");
43.        div.setBounds(360,250,100,50);
44.        //Add ActionListener to all buttons
45.        add.addActionListener(obj);
46.        sub.addActionListener(obj);
47.        mul.addActionListener(obj);
48.        div.addActionListener(obj);
49.        //Add the buttons to frame
50.        frame.add(add);
51.        frame.add(sub);
52.        frame.add(mul);
53.        frame.add(div);
54.        //Create output text field
55.        out = new JTextField();
56.        out.setBounds(150,350,200,100);
57.        frame.add(out);
58.        //Display frame
59.        frame.setVisible(true);
60.    }
61.    //Perform the respective operation
62.    public void actionPerformed(ActionEvent e)
63.    {
64.        //Get the button
65.        String button = e.getActionCommand();
66.        //Get the numbers
67.        double num1=Double.valueOf(text1.getText());
68.        double num2=Double.valueOf(text2.getText());
```

```

69. switch(button)
70. {
71.   case "Add" :
72.     out.setText(num1+" + "+num2+" = "+(num1+num2));
73.     break;
74.   case "Subtract" :
75.     out.setText(num1+" - "+num2+" = "+(num1-num2));
76.     break;
77.   case "Multiply" :
78.     out.setText(num1+" * "+num2+" = "+(num1*num2));
79.     break;
80.   case "Divide" :
81.     out.setText(num1+" / "+num2+" = "+(num1/num2));
82.   }
83. }
84.

```

Program Explanation

1. Add the components to the frame at appropriate position.
2. To get the label of button clicked, use **getActionCommand**.
3. Obtain the text entered in the input fields and convert to **double** data type using **Double.valueOf**.

advertisement

Runtime Test Cases

Here's the run time test cases to perform arithmetic operations.

Test case 1 – To perform Addition.

The screenshot shows a Java Swing application window titled "Aritmetic Operations". The window has a light gray header bar with the title and a close button. Below the header, there are two text input fields. The first field is labeled "First Number :" and contains the value "10.5". The second field is labeled "Second Number :" and also contains the value "20". At the bottom of the window, there are four buttons arranged horizontally: "Add", "Subtract", "Multiply", and "Divide". Below these buttons is a large text area that displays the result of the addition: "10.5 + 20.0 = 30.5".

Test case 2 – To perform Subtraction.

Arithmetic Operations

x

First Number :

10.5

Second Number :

20

Add

Subtract

Multiply

Divide

10.5 - 20.0 = -9.5

Test case 3 – To perform Multiplication.

Arithmetic Operations

x

First Number :

10.5

Second Number :

20

Add

Subtract

Multiply

Divide

10.5 * 20.0 = 210.0

advertisement

Test case 4 – To perform Division.

Arithmetic Operations

x

First Number :

10.5

Second Number :

20

Add

Subtract

Multiply

Divide

$10.5 / 20.0 = 0.525$

160. Java Program to Display String with Background Color as Cyan in a Frame

[« Prev](#)

[Next »](#)

This is a Java Program to Display String with Background Color as Cyan in a Frame

Problem Description

We have to write a program in Java such that it creates a frame with background color as cyan and a text with color red is written in the frame.

Expected Input and Output

For displaying string with background color as cyan, we have the following set of input and output.

advertisement

To Display the Frame :

On the execution of the program,
it is expected that the frame has background color as cyan, and also
displays the text "The Background of Frame is Cyan" in frame with red color.

Problem Solution

1. Create the frame.
2. Set the background color of frame as cyan using **setBackground(Color.CYAN)**.
3. Create a text with required font type and size.
4. Set the color of text to red using **setForeground(Color.red)**.
5. Display the frame.

advertisement

Program/Source Code

Here is source code of the Java Program to perform arithmetic operations. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Display Text with Background Color as Cyan*/
2. import javax.swing.*;
3. import java.awt.*;
4. class Background_Cyan
5. {
6.     //Driver function
7.     public static void main(String args[])
8.     {
9.         //Create a frame
10.        JFrame frame = new JFrame("Cyan Text");
11.        frame.setSize(400,400);
12.        frame.setLayout(new FlowLayout());
13.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14.        frame.getContentPane().setBackground(Color.CYAN);
15.        //Write some text to the frame
16.        String str="The Background of Frame is Cyan";
17.        JLabel text = new JLabel(str);
18.        text.setFont(new Font("Serif",Font.ITALIC,20));
19.        frame.add(text);
20.        //Set Background Color of Text as red
21.        text.setForeground(Color.red);
22.        //Display the frame
23.        frame.setVisible(true);
24.    }
```

Program Explanation

1. To set the background color of frame, use the **setBackground()** function.
2. To set the foreground color of text, use the **setForeground()** function.
advertisement

Runtime Test Cases

Here's the run time test cases to display text with background color as cyan.

Test case 1 – To view the frame.



161. Java Program to Display Human Face using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Display Human Face using Applet

Problem Description

We have to write a program in Java such that it displays a human face using applet.

Expected Input and Output

For displaying human face using applet, we can have the following set of input and output.

advertisement

When the Applet is Executed :

When the applet is executed,
it is expected that a human face is drawn.

Problem Solution

1. Initially, set the background color of frame as white.
2. Create a color which is similar to the human skin color, and then draw and fill the outer boundary of the face.
3. Then set the color to black.
4. Draw and fill the left eye and the right eye.
5. Draw the right eyebrow and the left eyebrow.
6. Draw the nose.
7. Draw the smile.

advertisement

Program/Source Code

Here is source code of the Java Program to draw a human face using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Draw a Human Face using Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. public class Human_Face extends Applet
5. {
6.     //Initialize the applet
7.     public void init()
8.     {
9.         setBackground(Color.white);
10.    }
11.    //Draw the human face
12.    public void paint(Graphics g)
13.    {
14.        //Change color to cream
15.        Color clr=new Color(255,179,86);
16.        g.setColor(clr);
17.        //Draw and fill the face
18.        g.drawOval(100,100,250,300);
19.        g.fillOval(100,100,250,300);
20.        //Change color to black
21.        g.setColor(Color.black);
22.        //Draw the left eye
```

```
23.     g.drawOval(160,185,40,25);
24.     g.fillOval(160,185,40,25);
25. //Draw the right eye
26.     g.drawOval(250,185,40,25);
27.     g.fillOval(250,185,40,25);
28. //Draw the Left Eyebrow
29.     g.drawArc(160,170,35,10,0,180);
30. //Draw the Right Eyebrow
31.     g.drawArc(250,170,35,10,0,180);
32. //Draw the Nose
33.     g.drawLine(210,265,210,275);
34.     g.drawLine(240,265,240,275);
35.     g.drawArc(210,275,30,10,0,-180);
36. //Draw the smile
37.     g.drawArc(175,300,100,50,0,-180);
38. }
39.*/
40.*/
41.<applet code = Human_Face.class width=500 height=500>
42.</applet>
43.*/
```

To execute the applet use the following commands:

advertisement

```
>>>javac Human_Face.java
>>>appletviewer Human_Face.java
```

Program Explanation

1. The RGB value (255,179,86) gives a kind of human skin color. Create a color using this value.
2. To draw and fill the face and eyes use **drawOval** and **fillOval** respectively.
3. To draw the eyebrows and smile use **drawArc**.
4. To draw the nose use **drawArc** and **drawLine**.

Runtime Test Cases

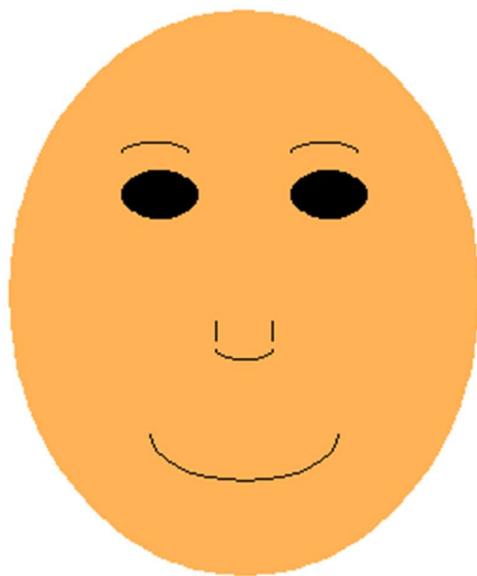
Here's the run time test cases for drawing human face using an applet.

advertisement

Test case 1 – To view the Human Face in an Applet.

Applet Viewer: Human_Face.class x

Applet



Applet started.

162. Java Program to Load and Display an Image Using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Load and Display an Image Using Applet

Problem Description

We have to write a program in Java such that it loads and displays an image using applet.

Expected Input and Output

For loading and displaying image, we can have the following set of input and output.

advertisement

When the Applet is Executed :

This image is available at the present working directory as “logo.jpeg”



When the applet is executed,
it is expected that the image is displayed in the frame.

Problem Solution

1. Use method **getImage** to load the image.
2. Use method **drawImage** to draw the image.

advertisement

Program/Source Code

Here is source code of the Java Program to load and display image using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Applet to Load and Display Image*/
2. import java.awt.*;
3. import java.applet.*;
4. public class Load_Image extends Applet
5. {
6.     Image image;
7.     //Function to Load the image
```

```

8. public void init()
9. {
10. image=getImage(getCodeBase(),"logo.jpeg");
11. }
12. //Function to draw the image
13. public void paint(Graphics g)
14. {
15. g.drawImage(image,0,0,this);
16. }
17. }
18.*/
19.<applet code = Load_Image.class width=500 height=500>
20.</applet>
21.*/

```

To execute the applet use the following commands:

advertisement

```
>>>javac Load_Image.java
>>>appletviewer Load_Image.java
```

Program Explanation

1. The method **getImage(getCodeBase(),image file name)** is used to get the image.
2. The method **drawImage(image file,x,y,component)** is used to draw the image starting from the x and y co-ordinates in the component specified.

advertisement

Runtime Test Cases

Here's the run time test cases to Load and Display Image using Applet.

Test case 1 – To Load and Display Image.



163. Java Program to Find Square of a Given Number Using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Find Square of a Given Number Using Applet

Problem Description

We have to write a program in Java such that it accepts a number and displays the square of the number using applet.

Expected Input and Output

For finding square of a number using applet, we can have the following different sets of input and output.

advertisement

1. To find the square of a number :

When the number 12 is entered,
it is expected that the square of 12 that is 144 is displayed in the frame.

2. To find the square of a floating-point number :

advertisement

When the number 10.5 is entered,
it is expected that the square of 10.5 that is 110.25 is displayed in the frame.

Problem Solution

1. Create an input text field to accept a number from user.
2. Create a button to confirm the number and calculate its square.
3. When the button is clicked, display its square in a text field.

Program/Source Code

Here is source code of the Java Program to find square of number using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Applet to Find Square of Number using Applet*/
2. import java.awt.*;
3. import java.applet.*;
4. import java.awt.event.*;
5. public class Square_Num extends Applet implements ActionListener
6. {
7.     TextField num,out;
8.     //Initialize with required input fields
9.     public void init()
10.    {
11.        Label label = new Label("Enter a number :");
12.        this.add(label);
13.        num = new TextField(5);
14.        this.add(num);
15.        Button calc = new Button("Calculate");
16.        calc.addActionListener(this);
17.        this.add(calc);
18.        out = new TextField();
19.    }
20.    @Override
```

```

21. //Function to display the square of the number
22. public void actionPerformed(ActionEvent e)
23. {
24.     double n = Double.valueOf(num.getText());
25.     double sq=n*n;
26.     out.setText("Square of "+n+" is "+sq);
27.     this.add(out);
28.     revalidate();
29. }
30.}
31.*/
32.<applet code = Square_Num.class width=500 height=500>
33.</applet>
34.*/

```

To execute the applet use the following commands:

```

>>>javac Square_Num.java
>>>appletviewer Square_Num.java

```

Program Explanation

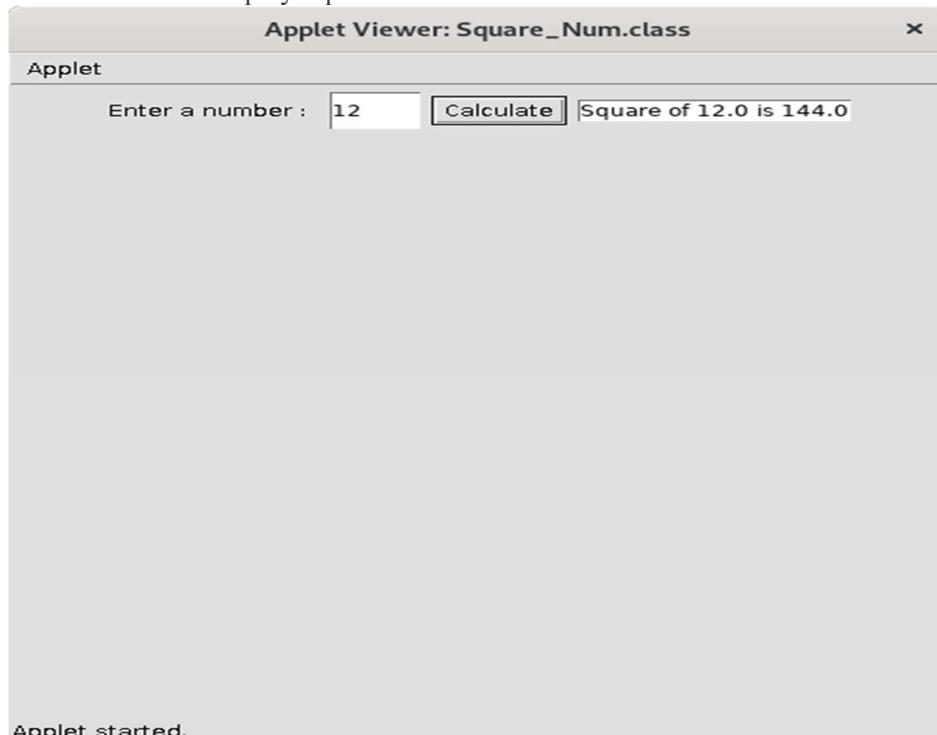
1. Create a text field using **TextField** class of the **java.awt** package.
2. Create a button using **Button** class of the **java.awt** package.
2. Create a label using **Label** class of the **java.awt** package.

advertisement

Runtime Test Cases

Here's the run time test cases to Display Square of Number using Applet.

Test case 1 – To Display Square of a Number.



Test case 2 – To Display Square of a floating-point number.



164. Java Program to Print Concentric Circles

« [Prev](#)

[Next](#) »

This is a Java Program to Print Concentric Circles

Problem Description

We have to write a program in Java such that it prints five concentric circles in an applet using **drawOval** function of **Graphics** class.

Expected Input and Output

For print concentric circles using applet, we can have the following set of input and output.

advertisement

To view the concentric circles :

On the execution of the program,
it is expected that five concentric circles are displayed in the applet.

Problem Solution

1. Consider a radius for the smallest circle.
2. Draw five concentric circles using **drawOval** function of **Graphics** class.

advertisement

Program/Source Code

Here is source code of the Java Program to print concentric circles using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Applet to Print Concentric Circles using Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. public class Concentric extends Applet
5. {
6.     //Initialize the applet
7.     public void init()
8.     {
9.         setBackground(Color.yellow);
10.    }
11.    //Draw cocentric circle
12.    public void paint(Graphics g)
13.    {
14.        g.setColor(Color.red);
15.        int rad=25;
16.        int dia=50;
17.        for(int i=0;i<5;i++)
18.        {
19.            g.drawOval(250-(i*rad),250-(i*rad),(i+1)*dia,(i+1)*dia);
20.        }
21.    }
22.}
23.*/
24.<applet code = Concentric.class width=500 height=500>
25.</applet>
26./*
```

To execute the applet use the following commands:

advertisement

```
>>>javac Concentric.java
>>>appletviewer Concentric.java
```

Program Explanation

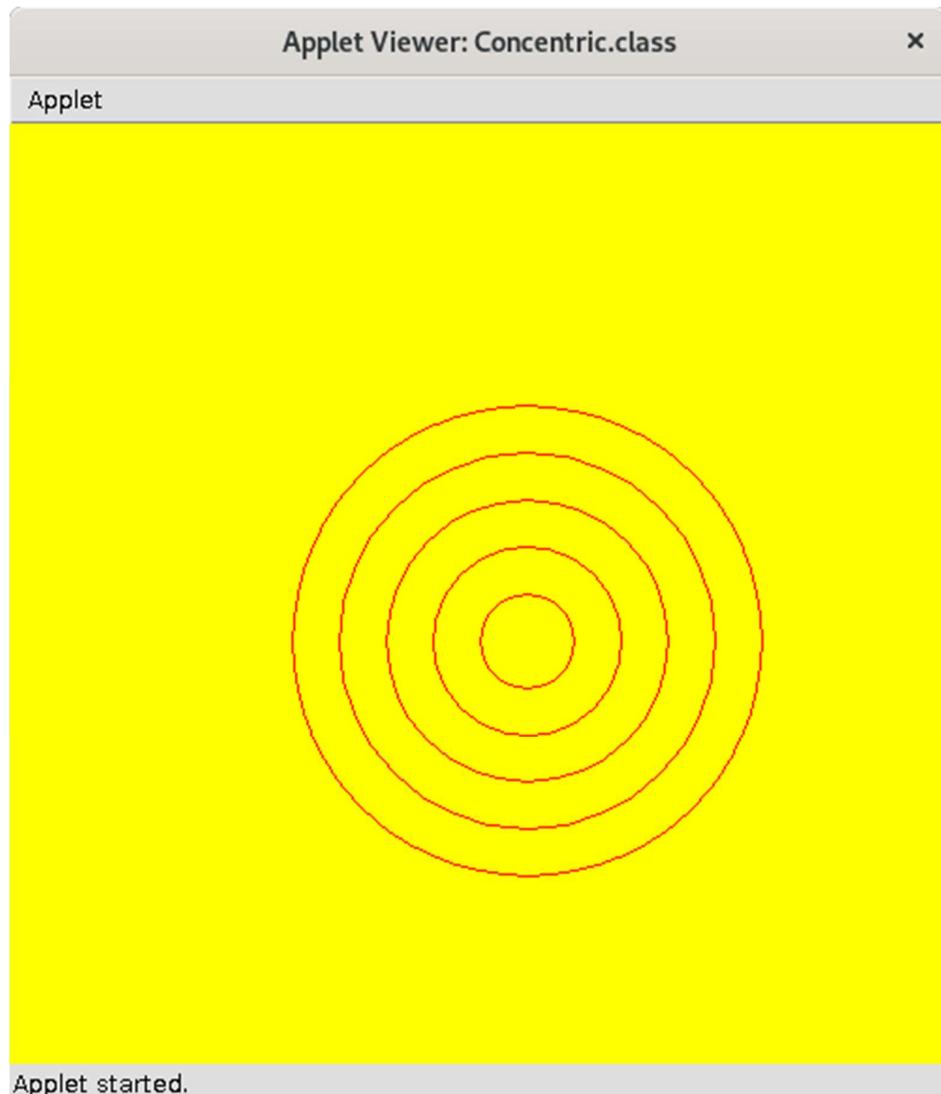
1. To draw concentric circles use the **drawOval(x,y,width,height)** to draw an circle starting from the (x,y) coordinate of equal height and width.
2. Increment/decrement the parameters of drawOval for each circle.

Runtime Test Cases

Here's the run time test cases to print concentric circles using Applet.

advertisement

Test case 1 – To Print Concentric Circles.



165. Java Program to illustrate the Use of Methods of Vector Class

[« Prev](#)

[Next »](#)

This is a Java Program to illustrate the Use of Methods of Vector Class

Problem Description

We have to write a program in Java such that it illustrates the use of methods of vector class using an applet.

Expected Input and Output

For illustrating methods of vector class, we can have the following different sets of input and output.

advertisement

1. To Add an Item to the Vector List:

When the user enters any item and selects the Add option,
then it is expected that the item gets added to the vector list.

For example, the vector list is [10, 20, 30, 40]

If you want to add a new item "15".

It is expected that 15 will be updated in vector list like [10, 20, 30, 40, 15]

If you want to add a new item "35".

It is expected that 15 will be updated in vector list like [10, 20, 30, 40, 15, 35]

2. To Delete an Item from the Vector List:

When the item is present in the vector.

When the user enters any item and selects the Delete option,
then it is expected that the item gets deleted from the vector list.

For example, the vector list is [10, 20, 30, 40, 15, 35]

Suppose if you want to delete "40" from the list.

It will delete and display message like "deleted successfully"

3. To Delete an Item from the Vector List:

When the item is not present in the vector.

advertisement

When the user enters any item and selects the Delete option,
then it is expected that appropriate error message is displayed.

For example, the vector list is [10, 20, 30, 15, 35]

Suppose if you want to delete "200" from the list.

If the number is not on the list. It displays a message like "could not be deleted"

4. To Search an Item in the Vector List:

When the item is present in the vector.

When the user enters any item and selects the Search option,
then it is expected that the index of item is displayed.

For example, the vector list is [10, 20, 30, 15, 35]

Suppose if you want to search the "15" element from the list.

Then expected output will be "15 found at index 3"

5. To Search an Item in the Vector List:

When the item is not present in the vector.

advertisement

When the user enters any item and selects the Search option,
then it is expected that appropriate error message is displayed.

For example, the vector list is [10, 20, 30, 15, 35]

Suppose if you want to search the "175" element from the list.

If the number is not on the list. It displays a message like "175 not found"

6. To Get the Capacity of the Vector List:

When the user selects the Capacity option,
then it is expected that the size of the vector list is displayed.

For example, the vector list is [10, 20, 30, 15, 35]

Then expected output will be "size of vector is 5"

Problem Solution

1. Create a vector and add few items to it.
2. Create two text field for displaying the vector and getting input from user respectively .
3. Create buttons for options – Add, Delete, Search and Capacity and add it to the frame.
4. Create a output text field.
5. Perform the operations using methods of the Vector class.

advertisement

Program/Source Code

Here is source code of the Java Program to illustrate the use of methods of vector class using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Illustrate the Use of Methods of Vector Class using Applet*/
2. import java.util.*;
3. import java.applet.*;
4. import java.awt.event.*;
5. import java.awt.*;
6. import javax.swing.*;
7. public class Vector_Class extends Applet implements ActionListener
8. {
9.     TextField num;
10.    Label list,out;
11.    Vector<Integer> vec;
12.    //Function to initialize the applet
13.    public void init()
14.    {
15.        setBackground(Color.white);
16.        setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));
17.        //Create a vector
18.        vec = new Vector<>();
19.        vec.add(10);
20.        vec.add(20);
21.        vec.add(30);
22.        vec.add(40);
23.    }
24.    //Function to set the features for applet
25.    public void start()
26.    {
27.        //Display the vector list
28.        list= new Label();
29.        list.setText("Vector List : "+vec);
30.        this.add(list);
31.        //Create a text field for number input
32.        num = new TextField("10");
33.        this.add(num);
34.        //Display the method options
35.        Button add = new Button("Add");
36.        Button del = new Button("Delete");
37.        Button search = new Button("Search");
38.        Button cap = new Button("Capacity");
39.        add.addActionListener(this);
40.        del.addActionListener(this);
41.        search.addActionListener(this);
42.        cap.addActionListener(this);
43.        this.add(add);
44.        this.add(del);
45.        this.add(search);
46.        this.add(cap);
47.        //Create the output field
48.        out = new Label();
```

```

49. this.add(out);
50. }
51. //Function to perform the selected option
52. public void actionPerformed(ActionEvent e)
53. {
54. String button = e.getActionCommand();
55. int number = Integer.valueOf(num.getText());
56. if(button.equals("Add"))
57. {
58. if(vec.add(number))
59.     out.setText(number+" added successfully");
60. else
61.     out.setText(number+" could not be added");
62. }
63. else if(button.equals("Delete"))
64. {
65. if(vec.removeElement(number))
66.     out.setText(number+" deleted successfully");
67. else
68.     out.setText(number+" could not be deleted");
69. }
70. else if(button.equals("Search"))
71. {
72. if(vec.contains(number))
73.     out.setText(number+" found at index "+vec.indexOf(number));
74. else
75.     out.setText(number+" not found");
76. }
77. else
78. {
79. out.setText("Size of vector is "+vec.size());
80. }
81. list.setText("Vector is : "+vec);
82. }
83.}
84.*/
85.<applet code = Vector_Class.class width=500 height=500>
86.</applet>
87.*/

```

To compile and run the applet use the following commands :

```

>>>javac Vector_Class.java
>>>appletviewer Vector_Class.java

```

Program Explanation

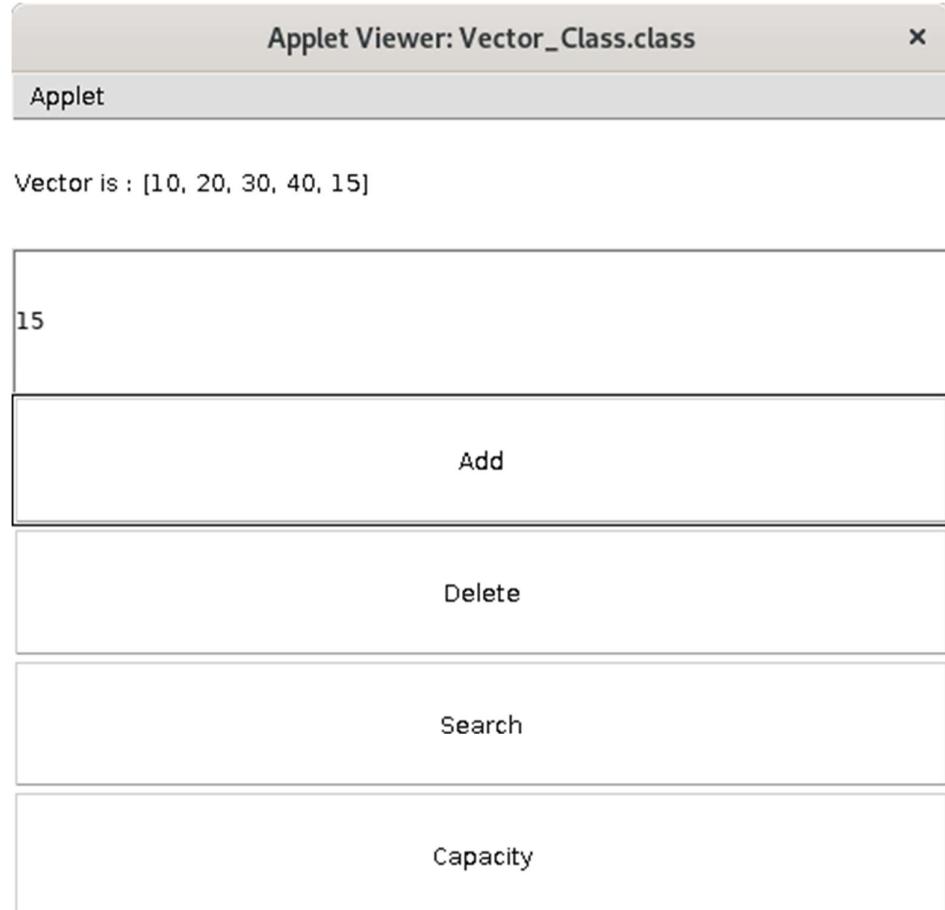
1. To create a vector use **Vector** and specify the type of data it contains.
2. To add an item to the vector use **(Vector Name).add(item)**.
3. To delete an item from the vector use **(Vector Name).removeElement(item)**.
4. To search an item in the vector use **(Vector Name).contains(item)**.
5. To get the capacity of vector use **(Vector Name).size()**. This function returns the size of the vector.
6. To get the index of an item in the vector use **(Vector Name).indexOf(item)**.

advertisement

Runtime Test Cases

Here's the run time test cases for illustrating the methods of vector class.

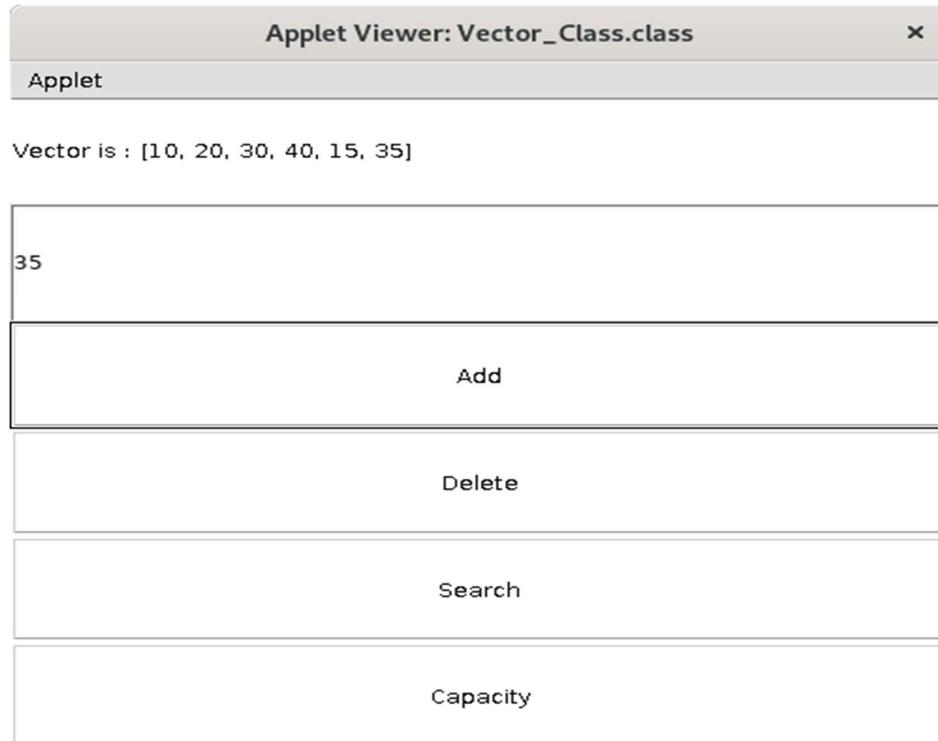
Test case 1 – To Add an Item to the Vector



15 added successfully

Applet started.

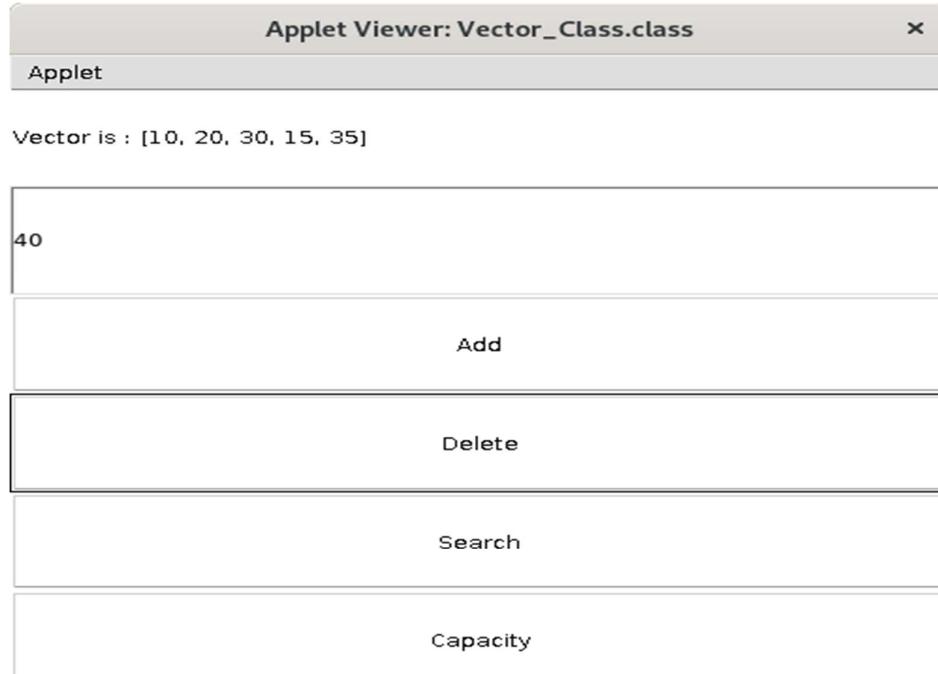
Test case 2 – To Add an Item to the Vector



35 added successfully

Applet started.

Test case 3 – To Delete an Item from the Vector



40 deleted successfully

Applet started.

Test case 4 – To Delete an Item from the Vector

Applet Viewer: Vector_Class.class

Applet

Vector is : [10, 20, 30, 15, 35]

200

Add

Delete

Search

Capacity

200 could not be deleted

Applet started.

Test case 5 – To Search an Item in the Vector

Applet Viewer: Vector_Class.class

Applet

Vector is : [10, 20, 30, 15, 35]

175

Add

Delete

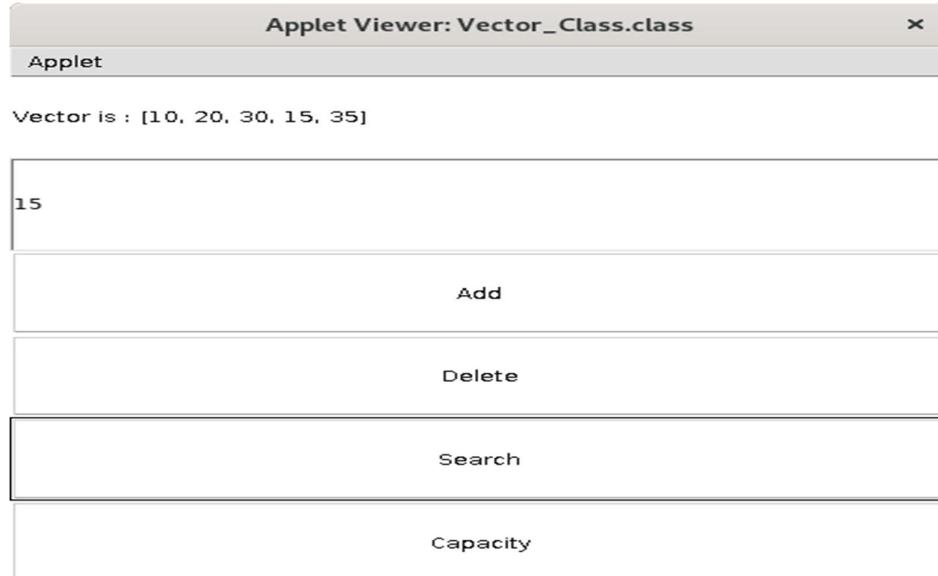
Search

Capacity

175 not found

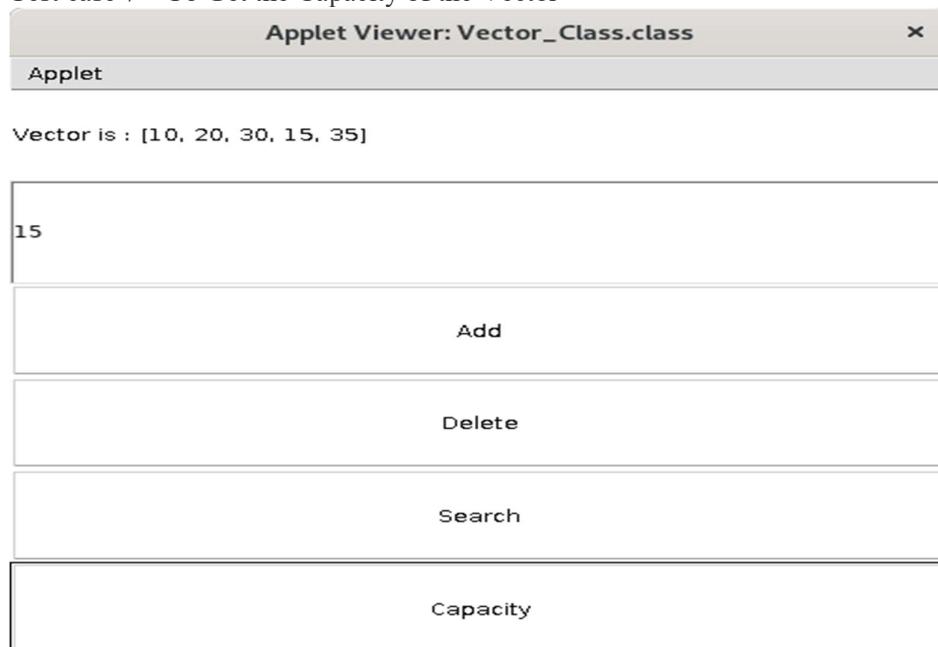
Applet started.

Test case 6 – To Search an Item in the Vector



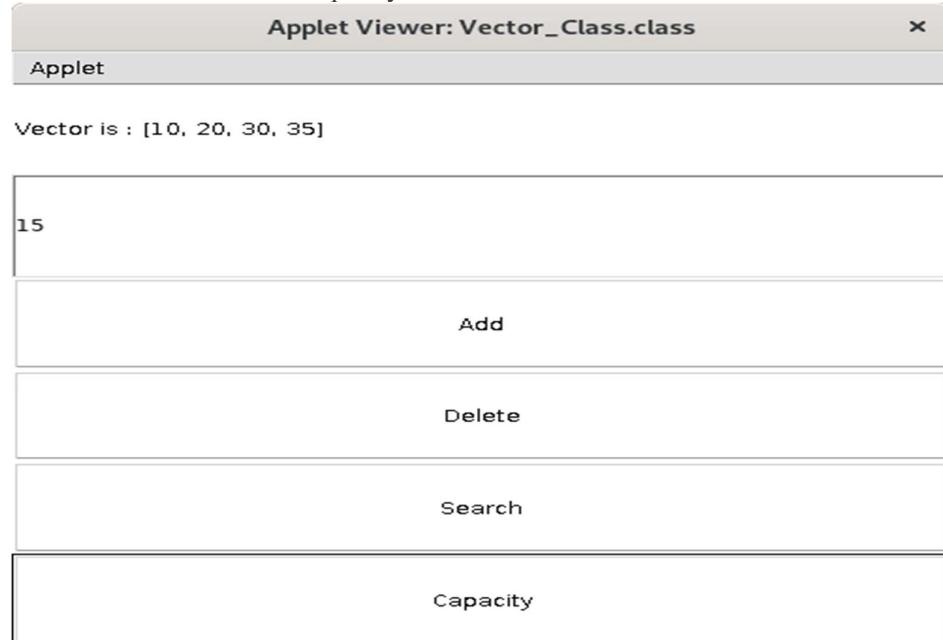
Applet started.

Test case 7 – To Get the Capacity of the Vector



Applet started.

Test case 8 – To Get the Capacity of the Vector



Size of vector is 4

Applet started.

166. Java Program to Demonstrate a Basic Calculator using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Demonstrate a Basic Calculator using Applet

Problem Description

We have to write a program in Java such that it creates a calculator which allows basic operations of addition, subtraction, multiplication and division.

Expected Input and Output

For creating a calculator, we can have the following different sets of input and output.

advertisement

1. To Perform Addition :

When the addition expression "98+102" is typed,
it is expected that the result is displayed as "98+102=200.0".

2. To Perform Subtraction :

When the subtraction expression "200.0-58.75" is typed,
it is expected that the result is displayed as "200.0-58.75=141.25".

3. To Perform Multiplication :

advertisement

When an multiplication expression "141.25*20" is typed,
it is expected that the result is displayed as "141.25*20=2825.0".

4. To Perform Division : When the denominator is non-zero

When an division expression "2825.0/5" is typed,
it is expected that the result is displayed as "2825.0/5=565.0".

5. To Perform Division : When the denominator is zero

advertisement

When an division expression "565.0/0" is typed,
it is expected that the error is displayed as "565.0/0=Zero Divison Error".

Problem Solution

1. Create a text field to accept the expression and display the output also.
2. Create buttons for digits and a decimal point.
3. Create a button to clear the complete expression.
4. Create the buttons for operations, that is for addition, subtraction, multiplication and division and an equals button to compute the result.
5. Add **ActionListener** to all the buttons.
6. Compute the result and display in the text field.

Program/Source Code

Here is source code of the Java Program to create a basic calculator. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Demonstrate a Basic Calculator using Applet*/
2. import java.awt.*;
3. import java.applet.*;
4. import java.awt.event.*;
5. public class Calculator extends Applet implements ActionListener
```

```

6. {
7.     TextField inp;
8.     //Function to add features to the frame
9.     public void init()
10.    {
11.        setBackground(Color.white);
12.        setLayout(null);
13.        int i;
14.        inp = new TextField();
15.        inp.setBounds(150,100,270,50);
16.        this.add(inp);
17.        Button button[] = new Button[10];
18.        for(i=0;i<10;i++)
19.        {
20.            button[i] = new Button(String.valueOf(9-i));
21.            button[i].setBounds(150+((i%3)*50),150+((i/3)*50),50,50);
22.            this.add(button[i]);
23.            button[i].addActionListener(this);
24.        }
25.        Button dec=new Button(".");
26.        dec.setBounds(200,300,50,50);
27.        this.add(dec);
28.        dec.addActionListener(this);
29.
30.        Button clr=new Button("C");
31.        clr.setBounds(250,300,50,50);
32.        this.add(clr);
33.        clr.addActionListener(this);
34.
35.        Button operator[] = new Button[5];
36.        operator[0]=new Button("/");
37.        operator[1]=new Button("*");
38.        operator[2]=new Button("-");
39.        operator[3]=new Button("+");
40.        operator[4]=new Button("=".equals(""));
41.        for(i=0;i<4;i++)
42.        {
43.            operator[i].setBounds(300,150+(i*50),50,50);
44.            this.add(operator[i]);
45.            operator[i].addActionListener(this);
46.        }
47.        operator[4].setBounds(350,300,70,50);
48.        this.add(operator[4]);
49.        operator[4].addActionListener(this);
50.    }
51.    String num1="";
52.    String op="";
53.    String num2="";
54.    //Function to calculate the expression
55.    public void actionPerformed(ActionEvent e)
56.    {
57.        String button = e.getActionCommand();
58.        char ch = button.charAt(0);
59.        if(ch>='0' && ch<='9'|| ch=='.')
60.        {
61.            if (!op.equals(""))
62.                num2 = num2 + button;
63.            else
64.                num1 = num1 + button;
65.            inp.setText(num1+op+num2);
66.        }
67.        else if(ch=='C')
68.        {
69.            num1 = op = num2 = "";
70.            inp.setText("");
71.        }

```

```

72. else if (ch =='=')
73. {
74.     if(!num1.equals("") && !num2.equals(""))
75.     {
76.         double temp;
77.         double n1=Double.parseDouble(num1);
78.         double n2=Double.parseDouble(num2);
79.         if(n2==0 && op.equals("/"))
80.         {
81.             inp.setText(num1+op+num2+" = Zero Division Error");
82.             num1 = op = num2 = "";
83.         }
84.         else
85.         {
86.             if (op.equals("+"))
87.                 temp = n1 + n2;
88.             else if (op.equals("-"))
89.                 temp = n1 - n2;
90.             else if (op.equals("/"))
91.                 temp = n1/n2;
92.             else
93.                 temp = n1*n2;
94.             inp.setText(num1+op+num2+" = "+temp);
95.             num1 = Double.toString(temp);
96.             op = num2 = "";
97.         }
98.     }
99.     else
100.    {
101.        num1 = op = num2 = "";
102.        inp.setText("");
103.    }
104. }
105. else
106. {
107.     if (op.equals("") || num2.equals(""))
108.         op = button;
109.     else
110.     {
111.         double temp;
112.         double n1=Double.parseDouble(num1);
113.         double n2=Double.parseDouble(num2);
114.         if(n2==0 && op.equals("/"))
115.         {
116.             inp.setText(num1+op+num2+" = Zero Division Error");
117.             num1 = op = num2 = "";
118.         }
119.         else
120.         {
121.             if (op.equals("+"))
122.                 temp = n1 + n2;
123.             else if (op.equals("-"))
124.                 temp = n1 - n2;
125.             else if (op.equals("/"))
126.                 temp = n1/n2;
127.             else
128.                 temp = n1*n2;
129.             num1 = Double.toString(temp);
130.             op = button;
131.             num2 = "";
132.         }
133.     }
134.     inp.setText(num1+op+num2);
135. }
136. }
137.

```

```
138./*
139.<applet code = Calculator.class width=600 height=600>
140.</applet>
141.*/
```

To compile and run the program use the following commands :

advertisement

```
>>>javac Calculator.java
>>>appletviewer Calculator.java
```

Program Explanation

1. When any button on the calculator is clicked, the function **actionPerformed** is called. Get the button clicked using **getActionCommand**.
2. If the button clicked is a digit or decimal point, then either of the following is executed :
 - a) The digit is concatenated to the the second number if no operator has been encountered.
 - b) Otherwise, the digit is concatenated to the first number.
3. If the button clicked is the clear button, then clear the input field.

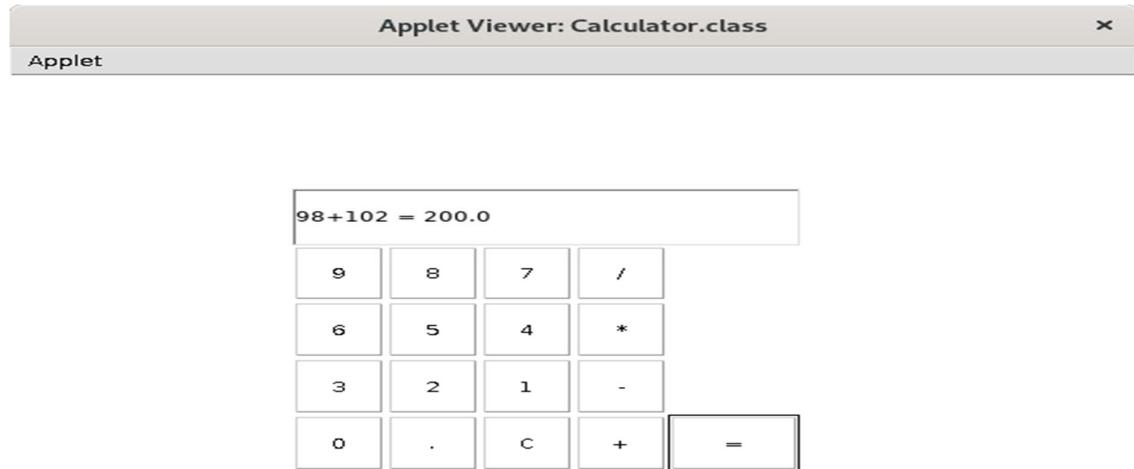
advertisement

4. If the button clicked is the equals operator, then either of the following is executed :
 - a) If neither the first number nor the second number is empty, then compute the result and display it in the frame
 - b) Otherwise, clear the input field.
5. If any of the button {+,-,*,/} is clicked, then either of the following is executed :
 - a) If either the operator or the second number is empty, then set the button clicked as the operator.
 - b) Otherwise, compute the result and display it in the frame.

Runtime Test Cases

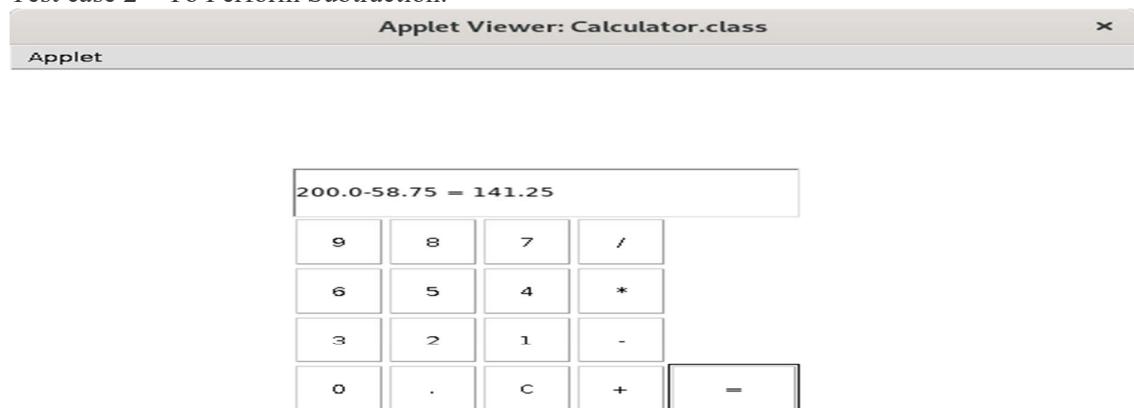
Here's the run time test cases for creating a basic calculator for different input cases.

Test case 1 – To Perform Addition.



Applet started.

Test case 2 – To Perform Subtraction.



Applet started.

Test case 3 – To Perform Multiplication.

Applet Viewer: Calculator.class

Applet

141.25*20 = 2825.0

9	8	7	/
6	5	4	*
3	2	1	-
0	.	c	+
=			

Applet started.

Test case 4 – To Perform Division by Non-Zero Value.

Applet Viewer: Calculator.class

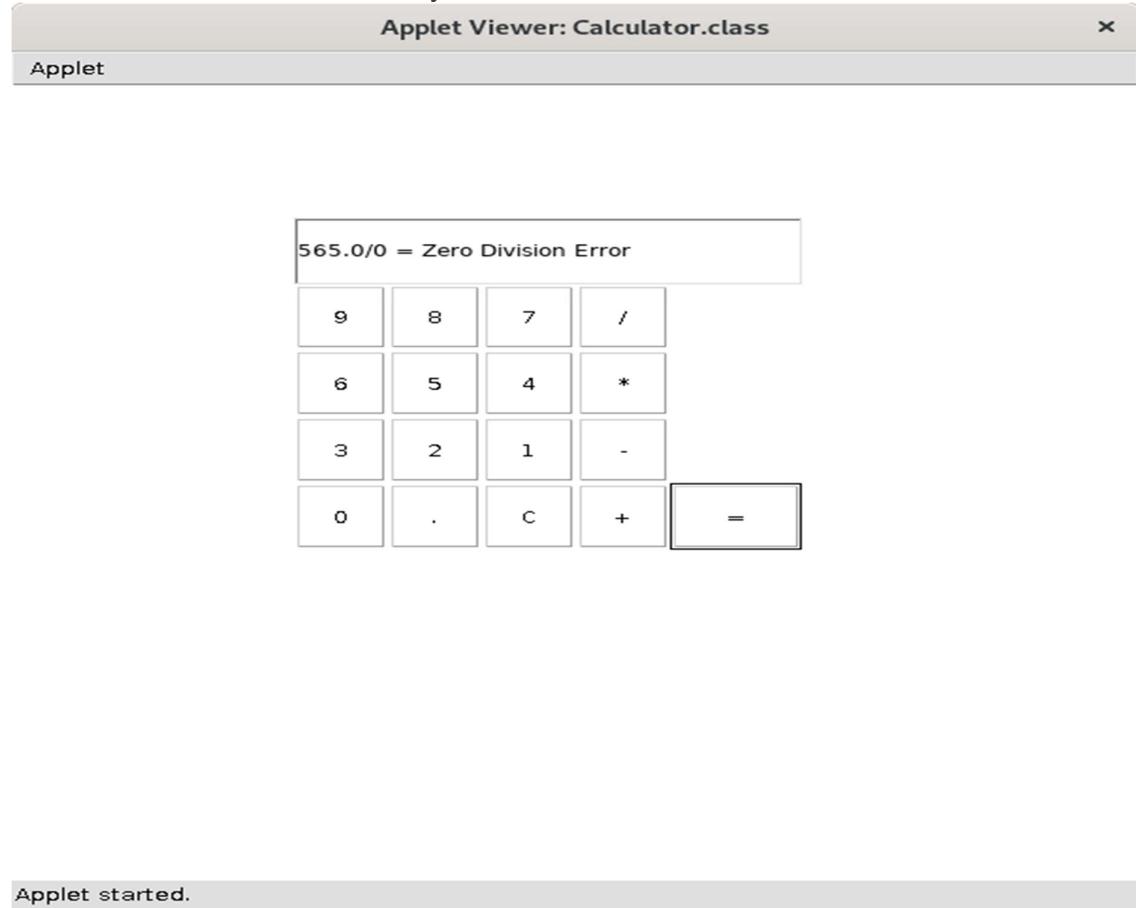
Applet

2825.0/5 = 565.0

9	8	7	/
6	5	4	*
3	2	1	-
0	.	c	+
=			

Applet started.

Test case 5 – To Perform Division by Zero.



167. Java Program to Create a Basic Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Create a Basic Applet

Problem Description

We have to write a program in Java such that it creates a basic applet with some text written on the applet.

Expected Input and Output

For creating a basic applet, we can have the following set of input and output.

advertisement

To View the Applet :

When the program is executed,
it is expected that an applet is created with "Hello World" written.

Problem Solution

1. To create an applet inherit the **Applet** class.
2. Write text on the frame, using **drawString** method of **Graphics** class.

advertisement

Program/Source Code

Here is source code of the Java Program to create a basic applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Create a Basic Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. public class Basic_Applet extends Applet
5. {
6.     //Function to initialize the applet
7.     public void init()
8.     {
9.         setBackground(Color.white);
10.        setLayout(new FlowLayout(FlowLayout.CENTER));
11.    }
12.    //Function to draw the string
13.    public void paint(Graphics g)
14.    {
15.        g.drawString("Hello World",100,200);
16.    }
17.}
18.*/
19.<applet code=Basic_Applet.class width=400 height=400>
20.</applet>
21.*
```

To compile and run the program use the following commands :

advertisement

```
>>>javac Basic_Applet.java
>>>appletviewer Basic_Applet.java
```

Program Explanation

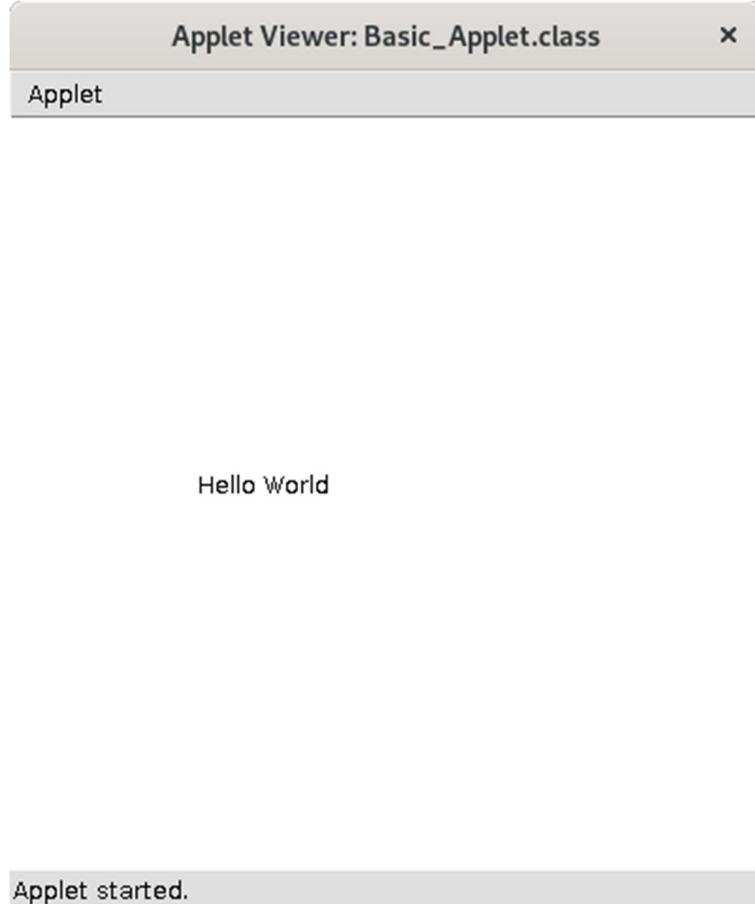
1. To create an applet, it is needed to inherit the **Applet** class.
2. The method **public void init()** is defined in the **Applet** class and it is used to initialize the applet.
3. The method **drawString(String,int x,int y)** draws a string starting from the (x,y) co-ordinate.

Runtime Test Cases

Here's the run time test cases for creating a basic applet for different input cases.

advertisement

Test case 1 – To View the Applet.



168. Java Program to Create and Fill Shapes using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Create and Fill Shapes using Applet

Problem Description

We have to write a program in Java such that it creates the shapes – Square, Pentagon, Circle, Oval, Rectangle and Triangle and fills color inside the shapes.

Expected Input and Output

For creating and filling shapes, we can have the following set of input and output.

advertisement

To View the Shapes :

When the program is executed,
it is expected that the applet consists of the following shapes
Square, Pentagon, Circle, Oval, Rectangle and Triangle
And also fills Yellow color inside the shapes.

Problem Solution

1. Use method **drawPolygon** and **fillPolygon** of the **Graphics** class to draw and fill the shapes – Square, Pentagon, Rectangle and Triangle.
2. Use method **drawOval** and **fillOval** of the **Graphics** class to draw and fill the shapes – Oval and Circle.

advertisement

Program/Source Code

Here is source code of the Java program to create and fill shapes. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Create and Fill Shapes using Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. public class Shapes extends Applet
5. {
6.     //Function to initialize the applet
7.     public void init()
8.     {
9.         setBackground(Color.white);
10.    }
11.    //Function to draw and fill shapes
12.    public void paint(Graphics g)
13.    {
14.        //Draw a square
15.        g.setColor(Color.black);
16.        g.drawString("Square",75,200);
17.        int x[]={50,150,150,50};
18.        int y[]={50,50,150,150};
19.        g.drawPolygon(x,y,4);
20.        g.setColor(Color.yellow);
21.        g.fillPolygon(x,y,4);
22.        //Draw a pentagon
23.        g.setColor(Color.black);
24.        g.drawString("Pentagon",225,200);
25.        x=new int[]{200,250,300,300,250,200};
```

```

26. y=new int[]{100,50,100,150,150,100};
27. g.drawPolygon(x,y,6);
28. g.setColor(Color.yellow);
29. g.fillPolygon(x,y,6);
30. //Draw a circle
31. g.setColor(Color.black);
32. g.drawString("Circle",400,200);
33. g.drawOval(350,50,125,125);
34. g.setColor(Color.yellow);
35. g.fillOval(350,50,125,125);
36. //Draw an oval
37. g.setColor(Color.black);
38. g.drawString("Oval",100,380);
39. g.drawOval(50,250,150,100);
40. g.setColor(Color.yellow);
41. g.fillOval(50,250,150,100);
42. //Draw a rectangle
43. g.setColor(Color.black);
44. g.drawString("Rectangle",300,380);
45. x=new int[]{250,450,450,250};
46. y=new int[]{250,250,350,350};
47. g.drawPolygon(x,y,4);
48. g.setColor(Color.yellow);
49. g.fillPolygon(x,y,4);
50. //Draw a triangle
51. g.setColor(Color.black);
52. g.drawString("Triangle",100,525);
53. x=new int[]{50,50,200};
54. y=new int[]{500,400,500};
55. g.drawPolygon(x,y,3);
56. g.setColor(Color.yellow);
57. g.fillPolygon(x,y,3);
58. }
59. }
60.*/
61.<applet code = Shapes.class width=600 height=600>
62.</applet>
63.*/

```

To compile and run the program use the following commands :

advertisement

```
>>>javac Shapes.java
>>>appletviewer Shapes.java
```

Program Explanation

1. The method **drawPolygon(int x[],int y[],int n)** is used to draw a polygon of n vertices, where each vertex has a co-ordinate (x,y) from the array of co-ordinates x[] and y[].
2. The method **fillPolygon(int x[],int y[],int n)** is used to fill a polygon of n vertices, where each vertex has a co-ordinate (x,y) from the array of co-ordinates x[] and y[].
3. The method **drawOval(int x,int y,int width,int height)** is used to draw an oval starting from the co-ordinate (x,y) of given width and height.

advertisement

4. The method **fillOval(int x,int y,int width,int height)** is used to fill an oval starting from the co-ordinate (x,y) of given width and height.

Runtime Test Cases

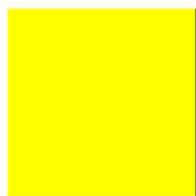
Here's the run time test cases for creating and filling shapes for different input cases.

Test case 1 – To View the Shapes.

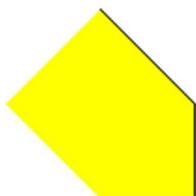
Applet Viewer: Shapes.class

x

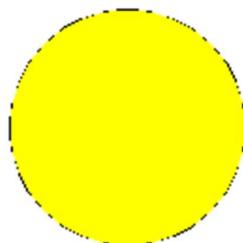
Applet



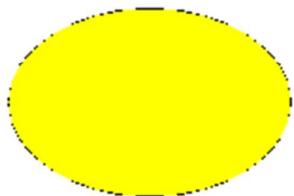
Square



Pentagon



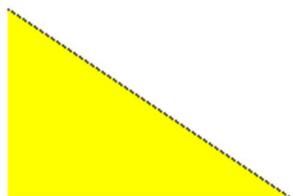
Circle



Oval



Rectangle



Traingle

Applet started.

169. Java Program to go to a Link using Applet

« [Prev](#)

[Next](#) »

This is a Java Program to go to a Link using Applet

Problem Description

We have to write a program in Java such that it creates an applet where the user has got choices of two links. When the user clicks on either of the buttons, the link is opened in the default web browser.

Expected Input and Output

For opening a link using applet, we can have the following 3 different sets of input and output.

advertisement

1. To view the applet :

When the program is executed,
it is expected that the applet contains two button - google and yahoo.

2. When the user selects Google :

If the user selects the button labeled "google",
then it is expected that the default google page is opened in the browser.

3. When the user selects Yahoo :

advertisement

If the user selects the button labeled "yahoo",
then it is expected that the default yahoo page is opened in the browser.

Problem Solution

1. Create an applet with the two buttons.
2. Add **ActionListener** to the buttons.
3. Form the complete URL of the page.
4. Use the **Desktop** class of **java.awt** package to open the link.

Program/Source Code

Here is source code of the Java Program to open a link using an applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /* Java Program to Go To a Link using Applet */
2. import java.awt.*;
3. import java.applet.*;
4. import java.net.*;
5. import java.awt.event.*;
6. import java.awt.Desktop.*;
7. public class Goto_Link extends Applet implements ActionListener
8. {
9.     //Function to initialize the applet
10.    public void init()
11.    {
12.        setBackground(Color.white);
13.        Button button1 = new Button("google");
14.        Button button2 = new Button("yahoo");
15.        this.add(button1);
16.        this.add(button2);
17.        button1.addActionListener(this);
18.        button2.addActionListener(this);
19.    }
```

```
20. //Function to go to the link
21. public void actionPerformed(ActionEvent e)
22. {
23. String button = e.getActionCommand();
24. String link = "http://www."+button+".com";
25. try
26. {
27. Desktop.getDesktop().browse(new URI(link));
28. }
29. catch (Exception E)
30. {
31. }
32. }
33.}
34.*/
35.<applet code=Goto_Link.class width=500 height=500>
36.</applet>
37.*/
```

To compile and execute the program use the following commands :

```
>>> javac Goto_Link.java
>>> appletviewer Goto_Link.java
```

Program Explanation

1. To form a complete link prefix “http://www.” before the domain and suffix “.com” after it.
2. Use **Desktop** class to handle the **URI (Uniform Resource Identifier)** file.
3. Use method **browse(URI)** to open the link in the default browser of the system.
advertisement

Runtime Test Cases

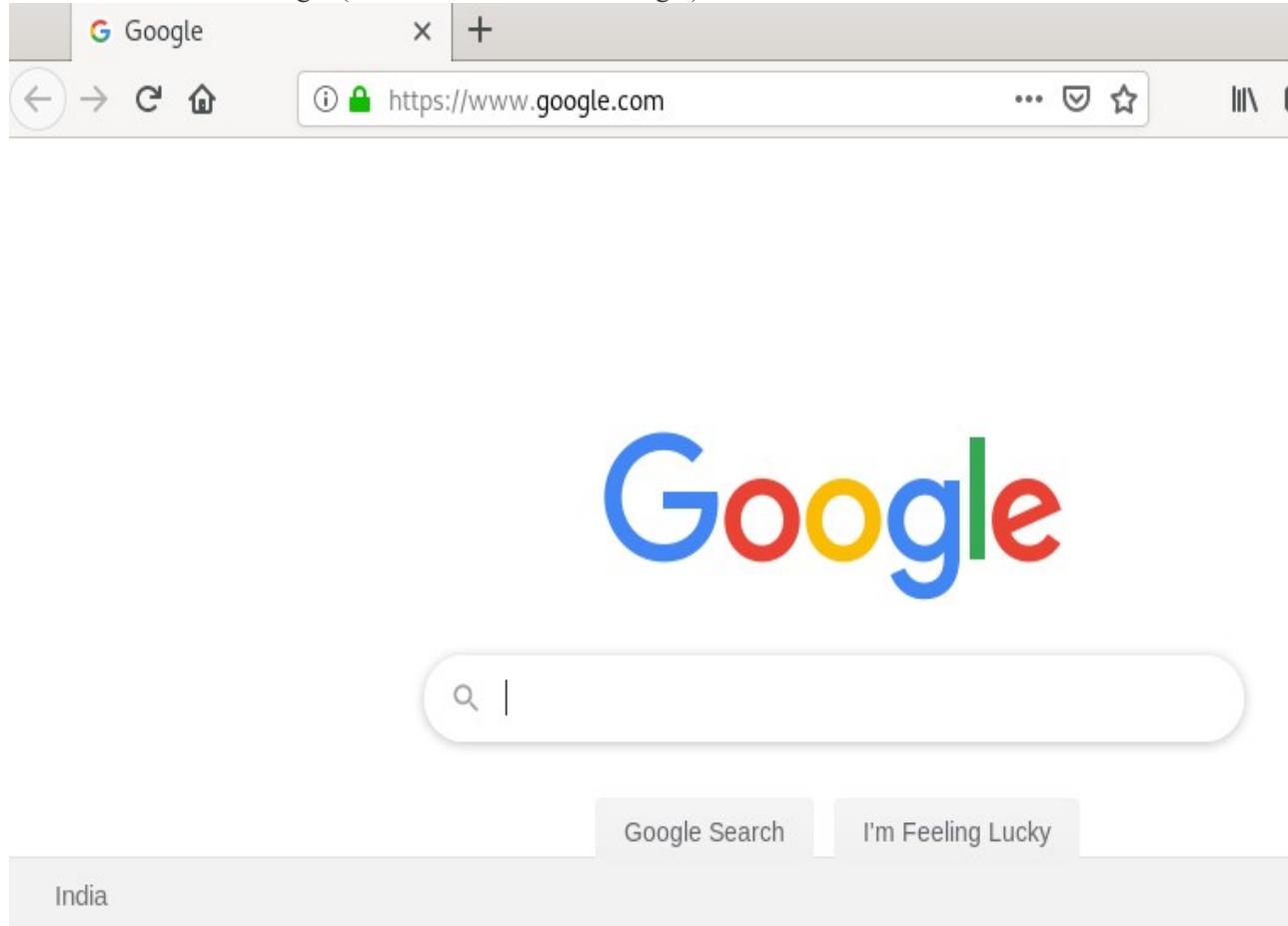
Here's the run time test cases for opening a link using an applet.

Test case 1 – To View the Applet

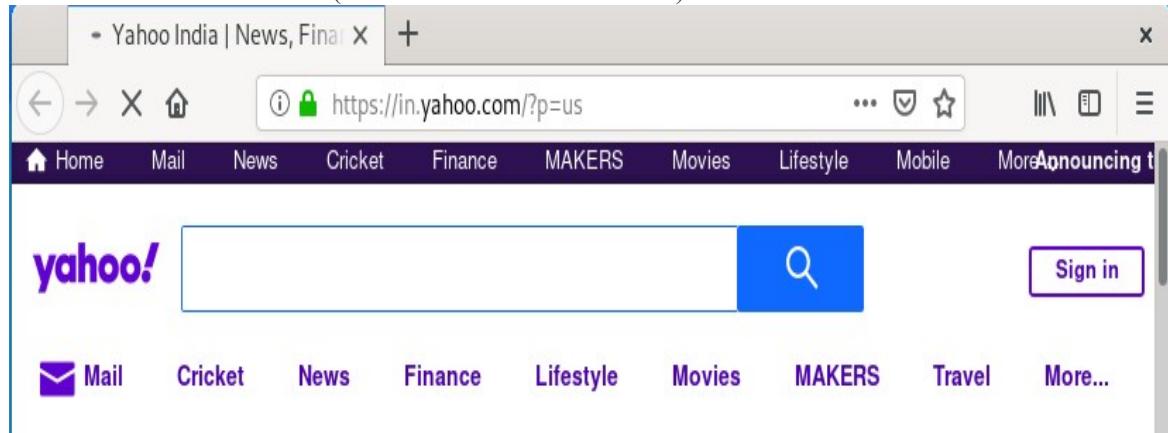


Applet started.

Test case 2 – Go To Google (When the user selects Google)



Test case 3 – Go To Yahoo (When the user selects Yahoo)



170. Java Program to Display a Clock Using Applet

« [Prev](#)

[Next](#) »

This is a Java Program to Display a Clock Using Applet

Problem Description

We have to write a program in Java such that it creates an analog clock which displays the current system time.

Expected Input and Output

For displaying a clock, we can have the following set of input and output.

advertisement

To View the Clock :

When the program is executed,
it is expected that the applet consists of a clock,
and the clock displays the current system time.

Problem Solution

1. Create a thread and delay each execution by 1 seconds.
2. Get the hour, minute and second parameters from the system time.
3. Create the clock and label the hours on it.
4. Draw the hours hand, minutes hand and the seconds hand based on the current system time.

advertisement

Program/Source Code

Here is source code of the Java program to display a clock. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Display a Clock using Applet*/
2. import java.applet.*;
3. import java.awt.*;
4. import java.util.*;
5. public class Clock extends Applet implements Runnable
6. {
7.     Thread t;
8.     //Initialize the applet
9.     public void init()
10.    {
11.        setBackground(Color.white);
12.    }
13.    //Function to start the thread
14.    public void start()
15.    {
16.        t = new Thread(this);
17.        t.start();
18.    }
19.    //Function to execute the thread
20.    public void run()
21.    {
22.        while(true)
23.        {
24.            try
25.            {
26.                repaint();
27.                //Delay by 1 sec
```

```

28.         Thread.sleep(1000);
29.     }
30.   catch(Exception e)
31.   {
32.   }
33. }
34. }
35. //Function to draw the clock
36. public void paint(Graphics g)
37. {
38.   Calendar time = Calendar.getInstance();
39.   int hour = time.get(Calendar.HOUR_OF_DAY) % 12;
40.   int minute = time.get(Calendar.MINUTE);
41.   int second = time.get(Calendar.SECOND);
42.   double angle;
43.   int x,y;
44.   //Draw a circle with center(250,250) & radius=150
45.   g.drawOval(100,100,300,300);
46.   //Label the clock
47.   String s="12";
48.   int i=0;
49.   while(i<12)
50.   {
51.     angle = Math.toRadians(30*(i-3));
52.     x = 250+(int)(Math.cos(angle)*135);
53.     y = 250+(int)(Math.sin(angle)*135);
54.     g.drawString(s,x,y);
55.     i++;
56.     s=String.valueOf(i);
57.   }
58.   //Draw the hours hand
59.   g.setColor(Color.green);
60.   angle = Math.toRadians((30*hour)-90);
61.   x = 250+(int)(Math.cos(angle)*100);
62.   y = 250+(int)(Math.sin(angle)*100);
63.   g.drawLine(250,250,x,y);
64.   //Draw the minutes hand
65.   g.setColor(Color.red);
66.   angle = Math.toRadians((6*minute)-90);
67.   x = 250+(int)(Math.cos(angle)*115);
68.   y = 250+(int)(Math.sin(angle)*115);
69.   g.drawLine(250,250,x,y);
70.   //Draw the seconds hand
71.   g.setColor(Color.blue);
72.   angle = Math.toRadians((6*second)-90);
73.   x = 250+(int)(Math.cos(angle)*130);
74.   y = 250+(int)(Math.sin(angle)*130);
75.   g.drawLine(250,250,x,y);
76. }
77. */
78./*
79.<applet code = Clock.class width=500 height=500>
80.</applet>
81.*/

```

To compile and run the program use the following commands :

advertisement

```
>>>javac Clock.java
>>>appletviewer Clock.java
```

Program Explanation

1. Use **Thread.sleep(int)** to delay the execution by entered time in milliseconds.
2. Create a circle for the clock and label the hours on it.

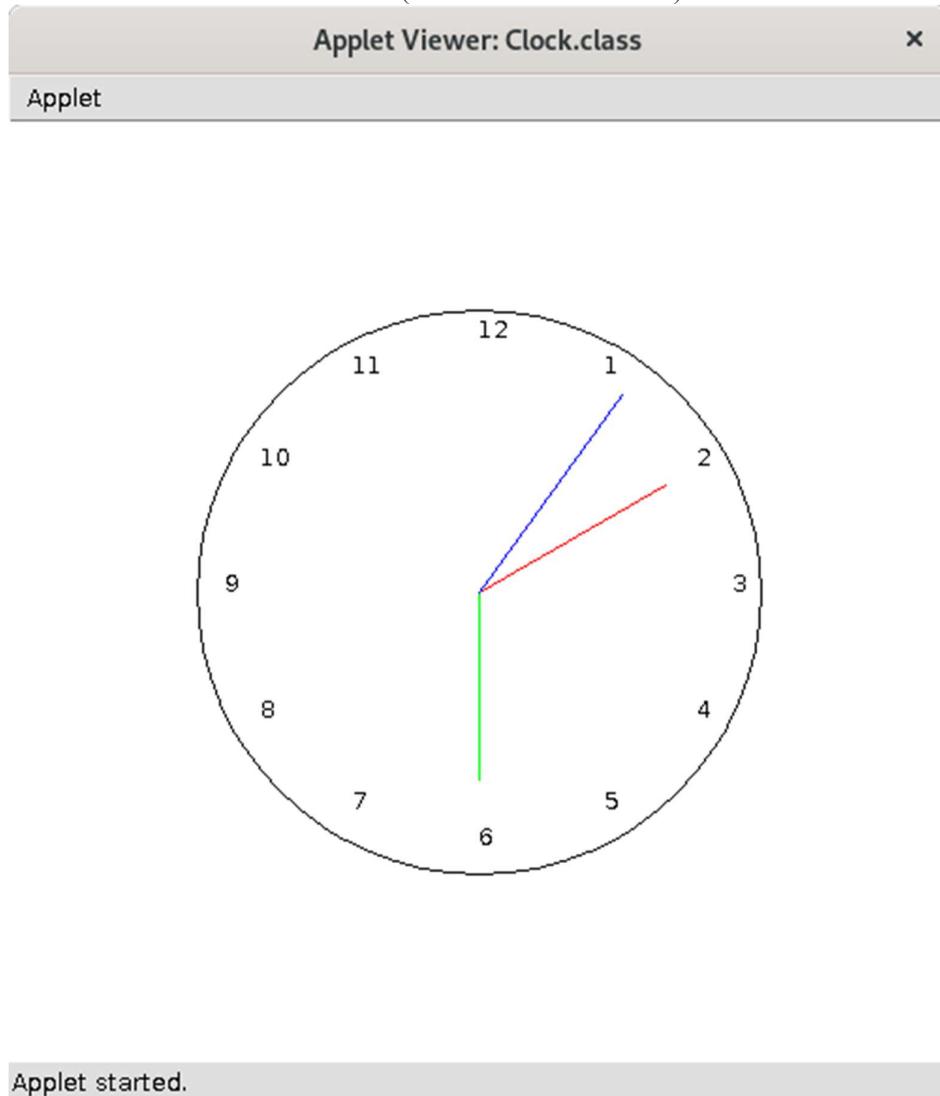
3. Each hour on the clock is equivalent to 30 degrees. The **angle of hours hand** from the $y=0$ line, is $((30 * \text{hour}) - 90)$ degrees.
4. Each minute on the clock is equivalent to 6 degrees. The **angle of minutes hand** from the $y=0$ line, is $((6 * \text{minute}) - 90)$ degrees.
5. Each second on the clock is equivalent to 6 degrees. The **angle of seconds hand** from the $y=0$ line, is $((6 * \text{second}) - 90)$ degrees.
6. To get the position of the hands of clock, it is required to convert the angles to radians. To do this, use **Math.toRadians** function.
7. The x co-ordinate of any point on a circle with center (a,b) and radius ' r ' measuring an angle θ from the center is given as $x = a + r * \cos\theta$.
8. The y co-ordinate of any point on a circle with center (a,b) and radius ' r ' measuring an angle θ from the center is given as $y = v + r * \sin\theta$.

Runtimetime Test Cases

Here's the run time test cases for creating a clock for different input cases.

advertisement

Test case 1 – To View the Clock. (When time is 18:10:06)

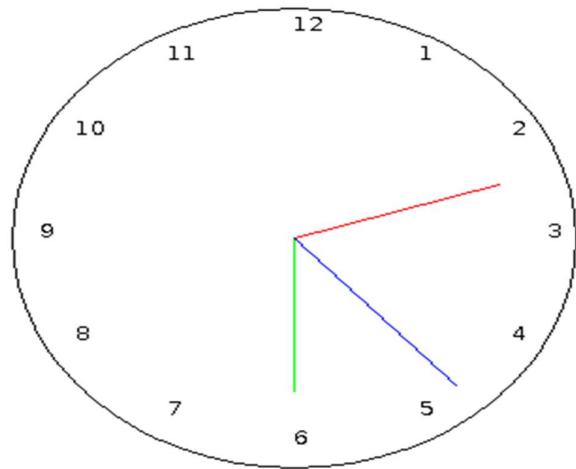


Test case 2 – To View the Clock. (When time is 18:12:23)

Applet Viewer: Clock.class

x

Applet



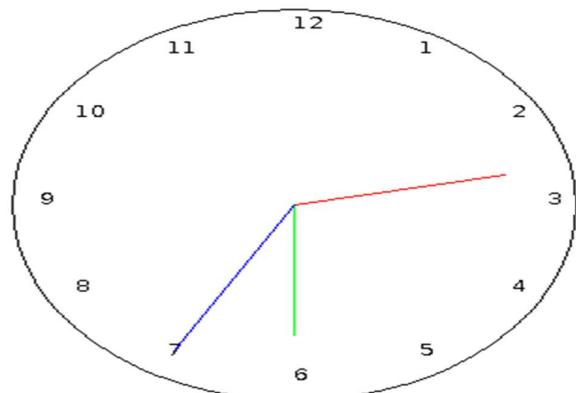
Applet started.

Test case 3 – To View the Clock. (When time is 18:14:35)

Applet Viewer: Clock.class

x

Applet



Applet started.

171. Java Program to Perform Read and Write Operations for a File using Applet

[« Prev](#)

[Next »](#)

This is a Java Program to Perform Read and Write Operations for a File using Applet

Problem Description

We have to write a program in Java such that it performs the read and write operations for a file using an applet.

Expected Input and Output

For performing read and write operations, we can have the following different sets of input and output.

advertisement

1. To Perform Read Operation : When the file is present in the directory.

When the user clicks on the read operation after entering the file name,
then it is expected that the contents of file is viewed in the applet.

2. To Perform Read Operation: When the file is not present in the directory.

When the user clicks on the read operation after entering the file name,
then it is expected that an appropriate message is displayed in the applet.

3. To Perform Write Operation: When the file is present in the directory.

advertisement

When the user clicks on the write operation after entering the file name,
then it is expected that the contents of the file is modified.

4. To Perform Write Operation: When the file is not present in the directory.

When the user clicks on the write operation after entering the file name,
then it is expected that a file is created with the contents.

Problem Solution

1. Create a text field to enter the file name.
2. Create two buttons for the read and write operations.
3. Create a text area when the contents of the file is displayed or written.
4. Use the **FileReader** class to read the file.
5. Use the **FileWriter** class to write to the file.

advertisement

Program/Source Code

Here is source code of the Java Program to perform read and write operations. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to Perform Read and Write Operations */
2. import java.applet.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. import java.io.*;
6. public class File extends Applet implements ActionListener
7. {
8.     TextField file;
9.     TextArea text;
10.    //Initialize the applet
11.    public void init()
12.    {
13.        setBackground(Color.white);
```

```
14. setLayout(null);
15.
16. Label label = new Label("File Name :");
17. label.setBounds(100,0,100,40);
18. this.add(label);
19.
20. file = new TextField();
21. file.setBounds(200,0,200,40);
22. this.add(file);
23.
24. text = new TextArea("",0,0,TextArea.SCROLLBARS_NONE);
25. text.setBounds(50,75,400,300);
26. this.add(text);
27.
28. Button read = new Button("Read From File");
29. read.setBounds(75,400,150,25);
30. this.add(read);
31. read.addActionListener(this);
32.
33. Button write = new Button("Write To File");
34. write.setBounds(275,400,150,25);
35. this.add(write);
36. write.addActionListener(this);
37. }
38. //Function to call functions for operations selected
39. public void actionPerformed(ActionEvent e)
40. {
41. String button = e.getActionCommand();
42. if(button.equals("Read From File"))
43. {
44. read();
45. }
46. else
47. {
48. write();
49. }
50. }
51. //Function to Read the File
52. public void read()
53. {
54. try
55. {
56. FileReader obj = new FileReader(file.getText());
57. int i;
58. text.setText("");
59. while((i=obj.read())!= -1)
60. {
61. text.setText(text.getText()+(char)i);
62. }
63. obj.close();
64. }
65. catch(Exception E)
66. {
67. text.setText(E.getMessage());
68. }
69. }
70. //Function to Write to the File
71. public void write()
72. {
73. try
74. {
75. FileWriter obj = new FileWriter(file.getText());
76. obj.write(text.getText());
77. obj.close();
78. text.setText("File Written Successfully");
79. }
```

```
80. catch(Exception E)
81. {
82.     text.setText(E.getMessage());
83. }
84. }
85.}
86.*/
87.<applet code = File.class width=500 height=500>
88.</applet>
89.*/
```

To compile and execute the program type the following commands :

advertisement

```
>>> javac File.java
>>> appletviewer File.java
```

Program Explanation

1. To read the file use **FileReader** class.
2. To write to the file use **FileWriter** class.

Runtime Test Cases

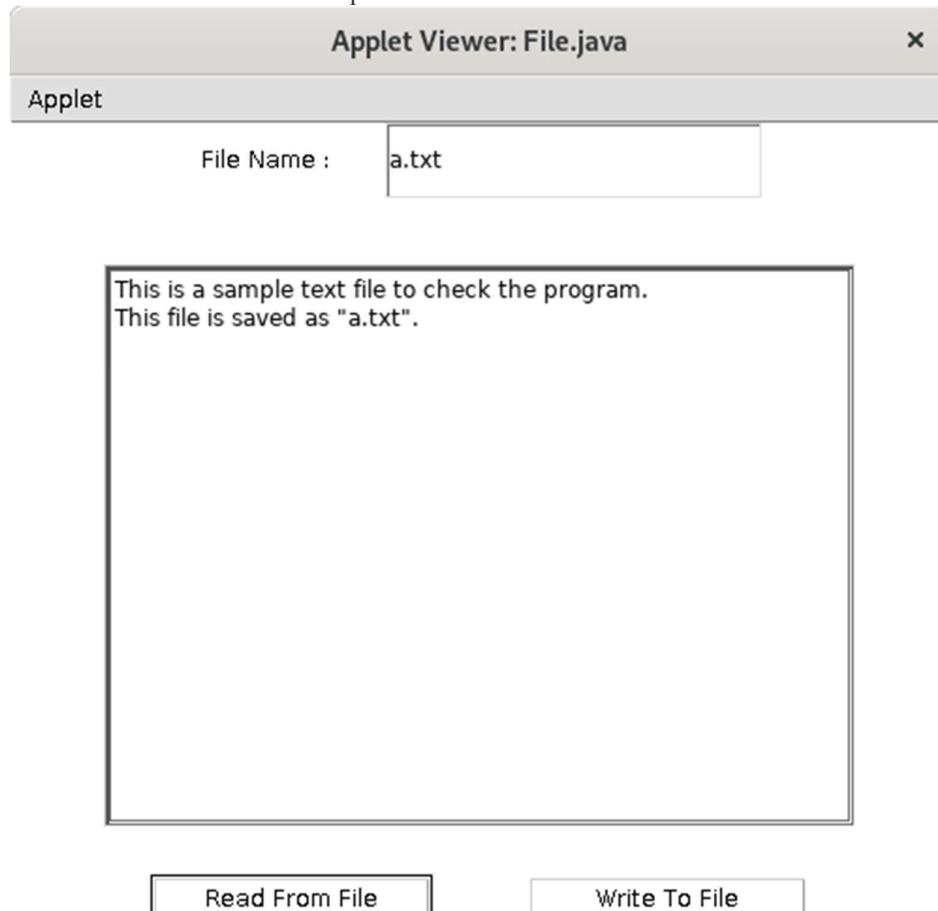
Here's the run time test cases for performing read and write for different input cases.

Test case 1 – To Test Write Operation.



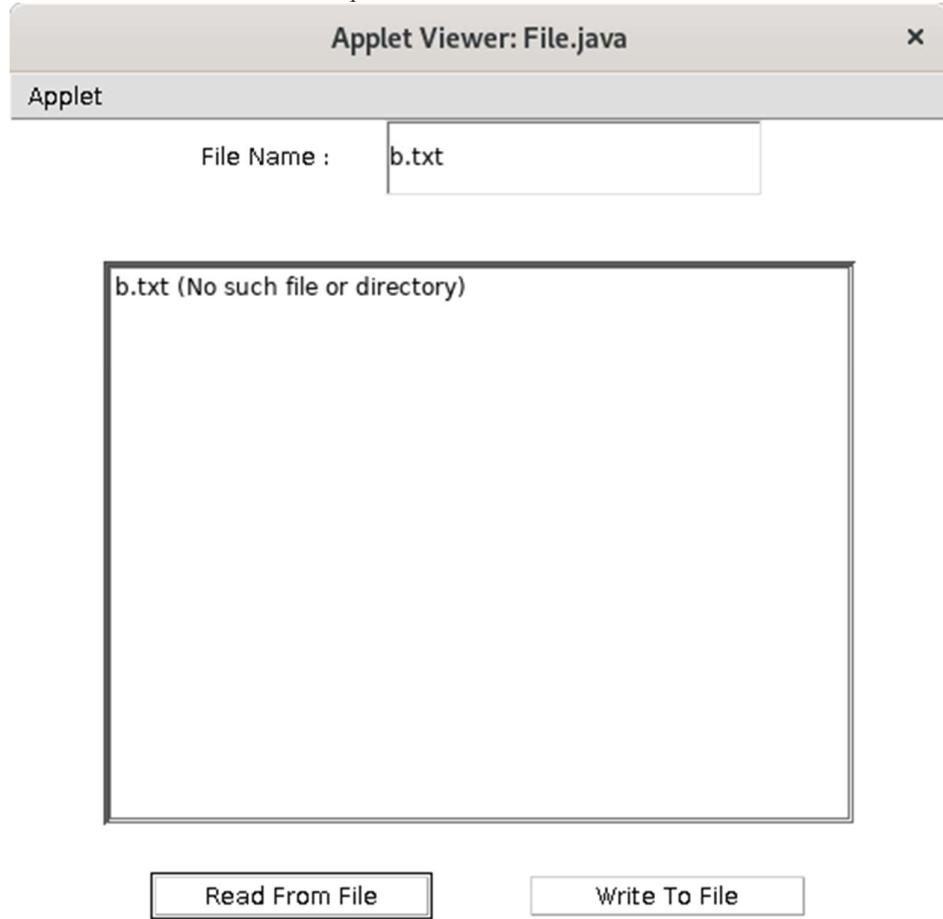
Applet started.

Test case 2 – To Test Read Operation When File is Present.



Applet started.

Test case 3 – To Test Read Operation When File is Not Present.



Applet started.

172. Java Program to Display Text in Different Fonts

[« Prev](#)

[Next »](#)

This is a Java Program to Display Text in Different Fonts

Problem Description

We have to write a program in Java such that it displays text in 10 different fonts.

Expected Input and Output

For displaying text in different fonts, we can have the following set of input and output.

advertisement

To View the Texts:

On every execution of the program,
it is expected that texts with different fonts are displayed.

Problem Solution

1. Get the list of all fonts available in the system using **GraphicsEnvironment**.
2. Create random fonts using the list generated.
3. Set the created font to the graphics object.
4. Draw the text using **drawString** method.

advertisement

Program/Source Code

Here is source code of the Java Program to display text in different fonts. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to Display Text in Different Fonts */
2. import java.applet.*;
3. import java.awt.*;
4. import java.awt.GraphicsEnvironment;
5. import java.lang.Math;
6. public class Text extends Applet
7. {
8.     String fonts[];
9.     //Function to get the list of fonts available
10.    public void init()
11.    {
12.        setBackground(Color.white);
13.
14.        GraphicsEnvironment GE;
15.        GE = GraphicsEnvironment.getLocalGraphicsEnvironment();
16.        fonts = GE.getAvailableFontFamilyNames();
17.    }
18.    //Function to draw the text
19.    public void paint(Graphics g)
20.    {
21.        int size = fonts.length;
22.        int pos;
23.        for(int i=1;i<=10;i++)
24.        {
25.            pos = (int)(Math.random()*size);
26.            Font my_font = new Font(fonts[pos],Font.PLAIN,20);
27.            g.setFont(my_font);
```

```
28.     g.drawString("Sample Text",100,(i*40));
29. }
30. }
31. }
32.*/
33.<applet code = Text.class width = 500 height = 600>
34.</applet>
35.*/
```

To compile and execute the program type the following commands :

advertisement

```
>>> javac Text.java
>>> appletviewer Text.java
```

Program Explanation

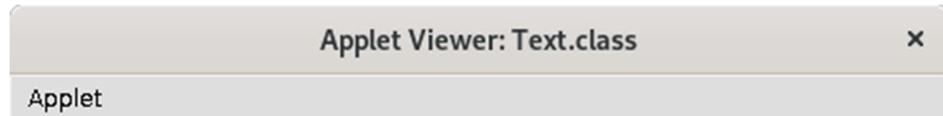
1. To get the local graphics environment, use the **getLocalGraphicsEnvironment** method.
2. To get the list of fonts, use **getAvailableFontFamilyNames** method.
3. Create a font using **Font** class with parameters font name, type of font and size of font.
4. Set the font to the graphics object using **setFont**.
5. Draw the string using **drawString**.

Runtime Test Cases

Here's the run time test cases to display text in different fonts for different input cases.

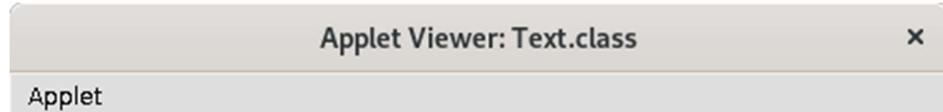
advertisement

Test case 1 – To View Text in Different Fonts.



Applet started.

Test case 2 – To View Text in Different Fonts.



Montserrat ExtraBold

cmmi10

Montserrat Black

□□□□□ □□□ □□□□□

Montserrat Light

LM Mono Prop Light 10

Cantarell Extra Bold

Padauk

□□□□□ □□□□□□□□

TeXGyreChorus

Applet started.

173. Java Program to Draw a Line Using GUI

[« Prev](#)

[Next »](#)

This is a Java Program to Draw a Line Using GUI

Problem Description

We have to write a program in Java such that it creates a line with random co-ordinates in an applet.

Expected Input and Output

For drawing a line, we can have the following set of input and output.

advertisement

To Draw a Line :

On every execution of the program,
it is expected that a line with random co-ordinates is drawn.

Problem Solution

1. Create a class and inherit the **Applet** class.
2. Generate four random integers x1, y1, x2, and y2 which lie within the bounds of the frame.
3. Draw a line from (x1,y1) to (x2,y2).

advertisement

Program/Source Code

Here is source code of the Java Program to draw a line. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to Draw a Line using GUI */
2. import java.applet.*;
3. import java.awt.*;
4. import java.lang.Math;
5. public class Line extends Applet
6. {
7.     //Function to initialize the applet
8.     public void init()
9.     {
10.         setBackground(Color.white);
11.     }
12.     //Function to draw the line
13.     public void paint(Graphics g)
14.     {
15.         int x1 = (int)(Math.random()*1000)%500;
16.         int y1 = (int)(Math.random()*1000)%500;
17.
18.         int x2 = (int)(Math.random()*1000)%500;
19.         int y2 = (int)(Math.random()*1000)%500;
20.
21.         g.drawLine(x1,y1,x2,y2);
22.     }
23. }
24.*/
25.<applet code = Line.class width = 500 height = 500>
26.</applet>
27.*/
```

To compile and execute the program use the following commands :

advertisement

```
>>> javac Line.java  
>>> appletviewer Line.java
```

Program Explanation

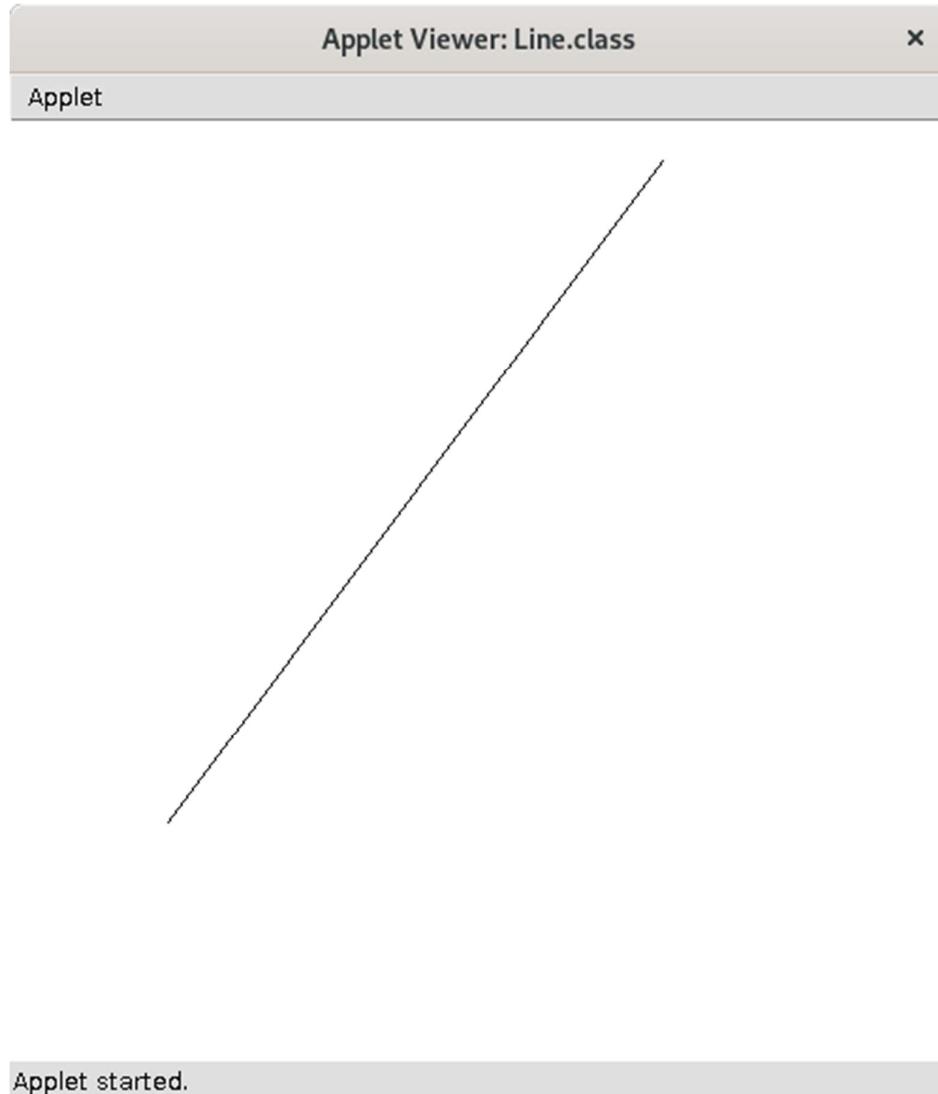
1. The method **Math.random** of the **Math** class generates a random number between 0 and 1.
2. The method **drawLine(x1,y1,x2,y2)** of the **Graphics** class is used to draw a line from (x1,y1) to (x2,y2).

Runtime Test Cases

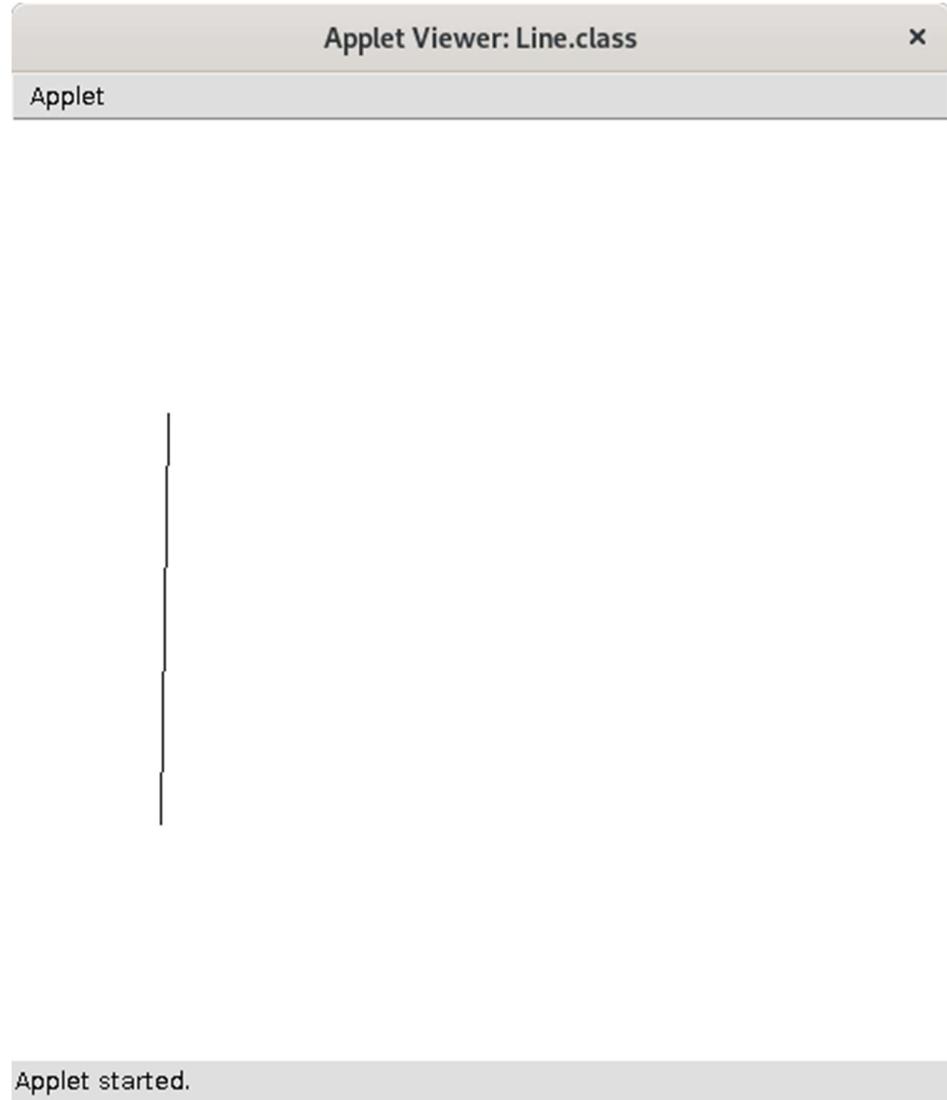
Here's the run time test cases to draw a line for different input cases.

advertisement

Test case 1 – To View a Line.



Test case 2 – To View a Line.



174. Java Program to Display a Message in a New Frame

[« Prev](#)

[Next »](#)

This is a Java Program to Display a Message in a New Frame

Problem Description

We have to write a program in Java such that it creates a frame with a button. When the button is clicked, a message is displayed in a new frame.

Expected Input and Output

For displaying a message in a new frame, we can have the following different sets of input and output.

advertisement

1. To View the Original Frame :

On every execution of the program,
it is expected that a frame with a button is created.

2. To View the Message in a New Frame :

advertisement

On click of the button,
it is expected that "!!! Hello !!!" message is displayed in a new frame.

Problem Solution

1. Create the original frame and add a button to it.
2. Add **ActionListener** to the button.
3. Display the original frame.
4. When the button is clicked, create a new frame and add a message to the frame.
5. Display the new frame.

Program/Source Code

Here is source code of the Java Program to display a message in a new frame. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /* Java Program to Display a Message in a New Frame */
2. import javax.swing.*;
3. import java.awt.event.*;
4. import java.awt.*;
5. class Message implements ActionListener
6. {
7.     //Function to create the original frame
8.     public static void main(String args[])
9.     {
10.        //Create a frame
11.        JFrame frame = new JFrame("Original Frame");
12.        frame.setSize(300,300);
13.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14.
15.        //Create an object
16.        Message obj = new Message();
17.
18.        //Create a button to view message
19.        JButton button = new JButton("View Message");
```

```

20. frame.add(button);
21. button.addActionListener(obj);
22.
23. //View the frame
24. frame.setVisible(true);
25. }
26. //Function to create a new frame with message
27. public void actionPerformed(ActionEvent e)
28. {
29. //Create a new frame
30. JFrame sub_frame = new JFrame("Sub Frame");
31. sub_frame.setSize(200,200);
32.
33. //Display the message
34. JLabel label = new JLabel("!!! Hello !!!");
35. sub_frame.add(label);
36.
37. //View the new frame
38. sub_frame.setVisible(true);
39. }
40.

```

Program Explanation

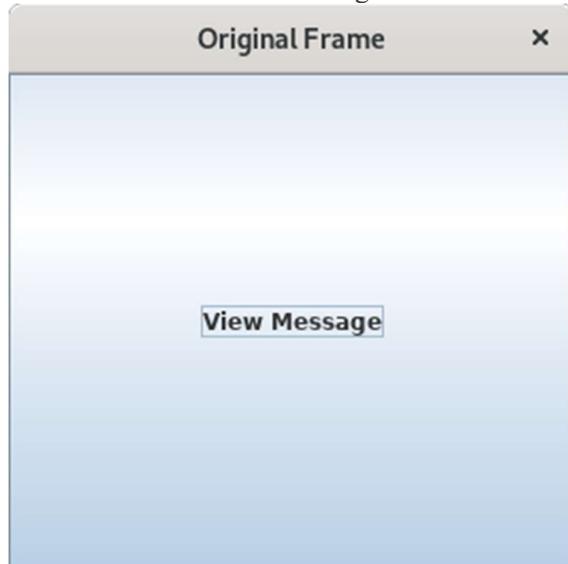
1. To create a frame, use the **JFrame** class.
2. To create a button, use the **JButton** class.
3. Add **ActionListener** to the button.

Runtime Test Cases

Here's the run time test cases to display message a in new frame for different input cases.

advertisement

Test case 1 – To View the Original Frame.



Test case 2 – To View the Message in New Frame.



175. Java Program to Display String in a Rectangle

« [Prev](#)

[Next](#) »

This is a Java Program to Display String in a Rectangle

Problem Description

We have to write a program in Java such that it creates a rectangle and displays a string inside it.

Expected Input and Output

For drawing string in a rectangle, we can have the following set of input and output.

advertisement

To View the String in a Rectangle :

On the execution of the program,
it is expected that a string is displayed inside a rectangle.

Given string is "String inside Rectangle".

Rectangle shape color should be "Red" and string color "Green".

Problem Solution

1. Draw a rectangle with required dimensions.
2. Create a string and draw it with proper co-ordinates.

advertisement

Program/Source Code

Here is source code of the Java Program to display string in a rectangle. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /* Java Program to Display String in a Rectangle */
2. import java.applet.*;
3. import java.awt.*;
4. public class String_Rectangle extends Applet
5. {
6.     //Initialize the applet
7.     public void init()
8.     {
9.         setBackground(Color.white);
10.    }
11.    //Function to draw rectangle and string
12.    public void paint(Graphics g)
13.    {
14.        g.setColor(Color.red);
15.
16.        int wid = 300;
17.        int len = 150;
18.        //Draw a rectangle
19.        g.drawRect(100,175,wid,len);
20.
21.        //Create a font and draw string
22.        Font myFont = new Font("TimesRoman",Font.BOLD,15);
23.        g.setFont(myFont);
24.        g.setColor(Color.green);
25.        String s = "String inside Rectangle";
26.        g.drawString(s,150,250);
27.    }
28.}
```

```
29./*
30.<applet code = String_Rectangle.class width = 500 height = 500>
31.</applet>
32.*/
```

To compile and execute the program use the following commands :

advertisement

```
>>> javac String_Rectangle.java
>>> appletviewer String_Rectangle.java
```

Program Explanation

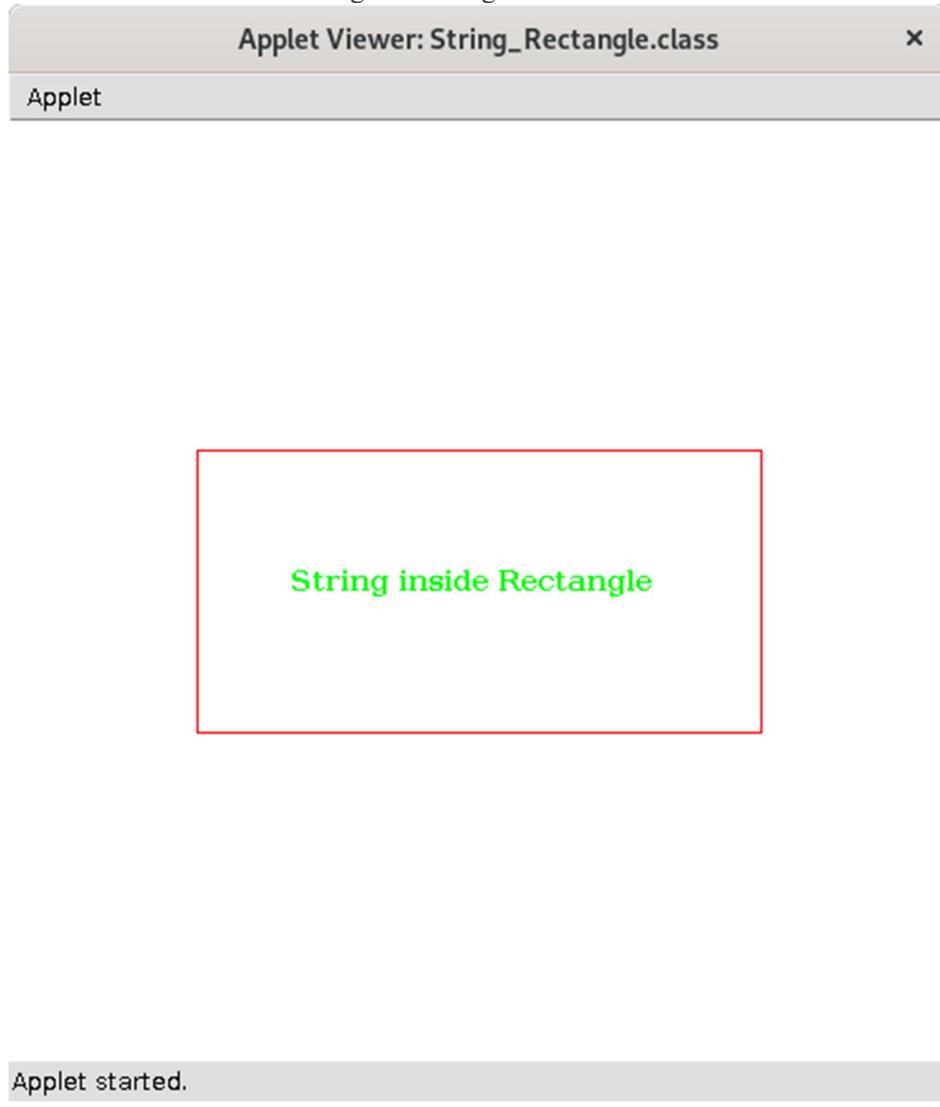
1. Use **drawRect(int x,int y,int width,int length)** method of **Graphics** class to draw a rectangle with dimensions width * length starting from (x,y) co-ordinate.
2. To draw a string use method **drawString**.

Runtime Test Cases

Here's the run time test cases to draw a string in rectangle for different input cases.

advertisement

Test case 1 – To View a String in Rectangle.



176. Java Program to Display a Pie Chart Using a Frame

[« Prev](#)

[Next »](#)

This is a Java Program to Display a Pie Chart Using a Frame

Problem Description

We have to write a program in Java such that it displays a pie chart for a given data

Expected Input and Output

For displaying a pie chart, we can have the following set of input and output.

advertisement

To View the Pie Chart :

When the program is executed,
it is expected that the frame contains a pie chart.

Problem Solution

1. Create an array of data type **int** to store the value of each data.
2. Create an array of data type **Color** to store the desired color in the pie chart for the respective data value.
3. Get the sum of all data values.
4. For each data value, calculate the percentage share in the pie chart.
5. Draw an arc for each data value.
6. Fill the arc with the respective color.

advertisement

Program/Source Code

Here is source code of the Java Program to create a pie chart. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Create a Pie Chart using a Frame */
2. import java.applet.*;
3. import java.awt.*;
4. public class Pie_Chart extends Applet
5. {
6.     int[] data_values;
7.     Color[] data_clr;
8.     int total;
9.     //Function to create a data set
10.    public void init()
11.    {
12.        setBackground(Color.white);
13.        //Create a data set to represent in pie-chart
14.        data_values=new int[]{10,25,12,15,15,18,5};
15.        data_clr=new Color[]{Color.red,Color.blue,Color.green,Color.yellow,
16.                           Color.orange,Color.black,Color.white};
17.    }
18.    //Function to get the sum of all data values
19.    public void start()
20.    {
21.        int n = data_values.length;
22.        int i;
23.        total=0;
24.        for(i=0;i<n;i++)
25.    {
```

```

26.     total+=data_values[i];
27. }
28. }
29. //Function to draw the pie chart
30. public void paint(Graphics g)
31. {
32. int i;
33. int start_angle = 0;
34. for(i=0;i<data_values.length;i++)
35. {
36.     int arc_angle = (int)(data_values[i]*360 / total);
37.     g.drawArc(100,100,300,300,start_angle,arc_angle);
38.     g.setColor(data_clr[i]);
39.     g.fillArc(100,100,300,300,start_angle,arc_angle);
40.     start_angle+=arc_angle;
41. }
42. }
43. }
44.*/
45.<applet code = Pie_Chart.class width=500 height=500>
46.</applet>
47.*/

```

To compile an execute the program use the following commands :

advertisement

```
>>>javac My_Cursor.java
>>>appletviewer My_Cursor.java
```

Program Explanation

1. Create arrays to store the data values and color for the respective data.
2. Get the sum of all data values.
3. Get the share of each data in the pie chart usin and draw an arc from the last position of arc to the calculated position.
4. Change the color of arc using **setColor** method.
5. Fill the arc using **fillArc** method.

Runtime Test Cases

Here's the run time test cases for creating a pie chart.

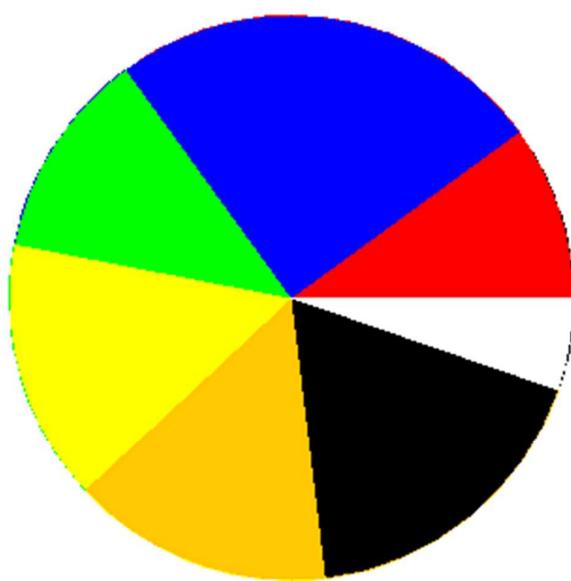
advertisement

Test case – To View the Pie Chart:

Applet Viewer: Pie_Chart.class

x

Applet



Applet started.

177. Java Program to Check Whether Antialiasing is Enabled or Not

[« Prev](#)

[Next »](#)

This is a Java Program to Check Whether Antialiasing is Enabled or Not

Problem Description

We have to write a program in Java such that it checks whether the antialiasing is enabled or not

Expected Input and Output

For checking whether antialiasing is enabled or not, we can have the following sets of input and output.

advertisement

1. When Antialiasing is Enabled :

If antialiasing is enabled,
it is expected that the message "Antialiasing is Enabled" is displayed.

2. When Antialiasing is Disabled :

advertisement

If antialiasing is disabled,
it is expected that the message "Antialiasing is Disabled" is displayed.

Problem Solution

1. Create a label with appropriate dimensions for the output message.
2. Convert the graphics object to **Graphics2D** type explicitly.
3. Create an object of class **RenderingHints** and get the attributes associated with the **Graphics2D** object.
4. Antialiasing is enabled if the object contains the value **VALUE_ANTIALIAS_ON**.
5. Display message accordingly.

Program/Source Code

Here is source code of the Java Program to check whether antialiasing is enabled or not. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /* Java Program to Check Whether Antialiasing is Enabled or Not */
2. import java.applet.*;
3. import java.awt.*;
4. import java.awt.RenderingHints;
5. public class Antialiasing extends Applet
6. {
7.     Label text;
8.     //Create the label
9.     public void init()
10.    {
11.        setBackground(Color.white);
12.        setLayout(null);
13.        text = new Label();
14.        text.setBounds(100,150,150,100);
15.        this.add(text);
16.    }
17.    //Function to check if antialiasing is enabled or not
18.    public void paint(Graphics g)
19.    {
20.        Graphics2D G = (Graphics2D)g;
```

```

21. RenderingHints rh = G.getRenderingHints();
22. if(rh.containsValue(RenderingHints.VALUE_ANTIALIAS_ON))
23.     text.setText("Antialiasing is Enabled");
24. else
25.     text.setText("Antialiasing is Disabled");
26. }
27.}
28.*/
29.<applet code = Antialiasing.class width=400 height=400>
30.</applet>
31.*/

```

To compile and execute the program use the following commands :

```

>>>javac Antialiasing.java
>>>appletviewer Antialiasing.java

```

Program Explanation

1. Explicitly type cast the object of **Graphics** class to **Graphics2D** type.
2. Use method **getRenderingHints()** to get the attributes of the **Graphics2D** object.
3. Use method **containsValue(parameter)** to check if the object contains the parameter. To check if antialiasing is enabled, the parameter is **RenderingHints.VALUE_ANTIALIAS_ON**.

advertisement

Runtime Test Cases

Here's the run time test cases for checking whether antialiasing is enabled or not for different input cases.

Test case 1 – If Antialiasing is Enabled:



Test case 2 – If Antialiasing is Disabled:

Applet Viewer: Antialiasing.class x

Applet

Antialiasing is Disabled

Applet started.

178. Java Program for Showing the Use of Various Methods of URL Class

[« Prev](#)

[Next »](#)

This is a Java Program for Showing the Use of Various Methods of URL Class

Problem Description

We have to write a program in Java such that it illustrates the use of methods of URL class using an applet.

Expected Input and Output

For illustrating methods of URL class, we can have the following different sets of input and output.

advertisement

1. To Get Protocol from the URL:

When the user enters an URL and selects the Get Protocol option, then it is expected that the protocol of the URL is displayed.

For example, given url is

"<https://www.google.com/search?client=firefox-b-d&q=java+founder>"
then it is expected that the protocol of the URL is "https"

2. To Get Host from the URL:

advertisement

When the user enters an URL and selects the Get Host option, then it is expected that the host of the URL is displayed.

For example, given url is

"<https://www.google.com/search?client=firefox-b-d&q=java+founder>"
then it is expected that the host of the URL is "www.google.com"

3. To Get Port from the URL:

When the user enters an URL and selects the Get Port option, then it is expected that the port of the URL is displayed.

advertisement

For example, given url is

"<https://www.google.com/search?client=firefox-b-d&q=java+founder>"
then it is expected that the port of the URL is -1

4. To Get Query from the URL:

When the user enters an URL and selects the Get Query option, then it is expected that the query string of the URL is displayed.

For example, given url is

"<https://www.google.com/search?client=firefox-b-d&q=java+founder>"
then it is expected that the query string of the URL is
"client=firefox-b-d&q=java+founder"

5. To Get File from the URL:

advertisement

When the user enters an URL and selects the Get File option, then it is expected that the file of the URL is displayed.

For example, given url is
"https://www.google.com/search?client=firefox-b-d&q=java+founder"
then it is expected that the file of the URL is
"File : /search?client=firefox-b-d&q=java+founder"

Problem Solution

1. Create a text field for the user to enter an URL address.
2. Create a text field to display the output.
3. Create buttons to perform the operations.
4. Perform the respective operations using the methods of URL class.

Program/Source Code

Here is source code of the Java Program to illustrate the use of methods of URL class using applet. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

```
1. /*Java Program to Illustrate the Use of Methods of URL Class using Applet*/
2. /*Java Applet to demonstrate the URL Class*/
3. import java.net.*;
4. import java.applet.*;
5. import java.awt.event.*;
6. import java.awt.*;
7. public class URL_Class extends Applet implements ActionListener
8. {
9.     TextField url_field;
10.    TextField out;
11.   //Function to initialize the applet
12.   public void init()
13.   {
14.       setBackground(Color.white);
15.       setLayout(null);
16.       //Create a label
17.       Label label = new Label();
18.       label.setText("Enter an URL : ");
19.       label.setBounds(50,50,100,50);
20.       this.add(label);
21.       //Create a text field
22.       url_field = new TextField();
23.       url_field.setBounds(150,50,250,50);
24.       this.add(url_field);
25.       //Create a textfield for output
26.       out = new TextField();
27.       out.setBounds(100,250,300,50);
28.       this.add(out);
29.   }
30.   //Function to set features to the applet
31.   public void start()
32.   {
33.       //Create buttons for method options
34.       Button protocol = new Button("Get Protocol");
35.       protocol.setBounds(0,150,100,50);
36.       protocol.addActionListener(this);
37.       this.add(protocol);
38.       Button host = new Button("Get Host");
39.       host.setBounds(100,150,100,50);
40.       host.addActionListener(this);
41.       this.add(host);
42.       Button port = new Button("Get Port");
43.       port.setBounds(200,150,100,50);
44.       port.addActionListener(this);
45.       this.add(port);
46.       Button query = new Button("Get Query");
47.       query.setBounds(300,150,100,50);
```

```

48. query.addActionListener(this);
49. this.add(query);
50. Button file = new Button("Get File");
51. file.setBounds(400,150,100,50);
52. file.addActionListener(this);
53. this.add(file);
54. }
55. //Function to perform the option selected
56. public void actionPerformed(ActionEvent e)
57. {
58. String button = e.getActionCommand();
59. try
60. {
61. URL url = new URL(url_field.getText());
62. if(button.equals("Get Protocol"))
63.     out.setText("Protocol : "+url.getProtocol());
64.
65. else if(button.equals("Get Host"))
66.     out.setText("Host :" +url.getHost());
67.
68. else if(button.equals("Get Port"))
69.     out.setText("Port : " +url.getPort());
70.
71. else if(button.equals("Get Query"))
72.     out.setText("Query String : " +url.getQuery());
73.
74. else
75.     out.setText("File : " +url.getFile());
76. }
77. catch(Exception exc)
78. {
79.     out.setText(exc.getMessage());
80. }
81. }
82. }
83.*/
84.<applet code = URL_Class.class width=500 height=500>
85.</applet>
86.*/

```

To compile and run the applet use the following commands :

advertisement

```
>>>javac URL_Class.java
>>>appletviewer URL_Class.java
```

Program Explanation

1. To create an URL use the **URL** class.
2. To get the protocol of the URL use **getProtocol** method.
3. To get the host of the URL use **getHost** method.
4. To get the port of the URL use **getPort** method.
5. To get the query string of the URL use **getQuery** method.
6. To get the file of the URL use **getFile** method.

Runtime Test Cases

Here's the run time test cases for illustrating the methods of URL class.

Test case 1 – To Get the Protocol of the URL

Applet Viewer: URL_Class.class

X

Applet

Enter an URL :

Get Protocol

Get Host

Get Port

Get Query

Get File

Protocol : https

Applet started.

Test case 2 – To Get the Host of the URL

Applet Viewer: URL_Class.class X

Applet

Enter an URL :

Host :www.google.com

Applet started.

Test case 3 – To Get the Port of the URL

Applet Viewer: URL_Class.class X

Applet

Enter an URL :

Applet started.

Test case 4 – To Get the Query String of the URL

Applet Viewer: URL_Class.class

x

Applet

Enter an URL :

Get Protocol

Get Host

Get Port

Get Query

Get File

Query String : client=firefox-b-d&q=java+founde

Applet started.

Test case 5 – To Get the File of the URL

Applet Viewer: URL_Class.class X

Applet

Enter an URL :

File : /search?client=firefox-b-d&q=java+founder

Applet started.

179. Java Program to Create a Color Dialog Box to Change the Background Color of Frame

« Prev

This is a Java Program to Create a Color Dialog Box to Change the Background Color of Frame

Problem Description

We have to write a program in Java such that it creates a button to open a color dialog box from where the user is able to select any color and the selected color is set as the background color of the frame.

Expected Input and Output

For changing color of frame using a color dialog box, we have the following different sets of input and output.

advertisement

1. When the frame is created :

On the execution of the program,
it is expected that a frame appears with a button to open the color dialog box.

2. When a color is chosen :

advertisement

When the button is clicked,
it is expected that a color dialog box appears and the user can select a color,
and the selected color becomes the background color of the frame.
suppose if we select the color Yellow.
then it is expected that the background color of the frame change to Yellow.

Problem Solution

1. Create a frame and a button.
2. Add **ActionListener** with the button.
3. When the button is clicked, create a color dialog box.
4. Get the selected color and change the background color of the frame.

Program/Source Code

Here is source code of the Java Program to change color of frame using color dialog box. The program is successfully compiled and tested using javac compiler on Fedora 30. The program output is also shown below.

advertisement

```
1. /*Java Program to change color of frame using color dialog box*/
2. import javax.swing.*;
3. import java.awt.*;
4. import java.awt.event.*;
5. class Frame_Color implements ActionListener
6. {
7.     static JFrame frame;
8.     //Driver function
9.     public static void main(String args[])
10.    {
11.        //Create a frame
12.        frame = new JFrame("Change Frame Background");
13.        frame.setSize(400,400);
14.        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15.        frame.getContentPane().setBackground(Color.white);
16.        frame.setLayout(new FlowLayout());
17.        //Create an object
18.        Frame_Color obj = new Frame_Color();
```

```

19. //Create a button
20. JButton button = new JButton("Change Color");
21. button.addActionListener(obj);
22. frame.add(button);
23. //Display the fame
24. frame.setVisible(true);
25. }
26. //Function to create color dialog box and change color
27. public void actionPerformed(ActionEvent e)
28. {
29. //Create a color dialog box
30. JColorChooser color_box= new JColorChooser();
31. Color color=color_box.showDialog(frame,"Select a Color",Color.white);
32. //Change background color of frame
33. frame.getContentPane().setBackground(color);
34. }
35.

```

Program Explanation

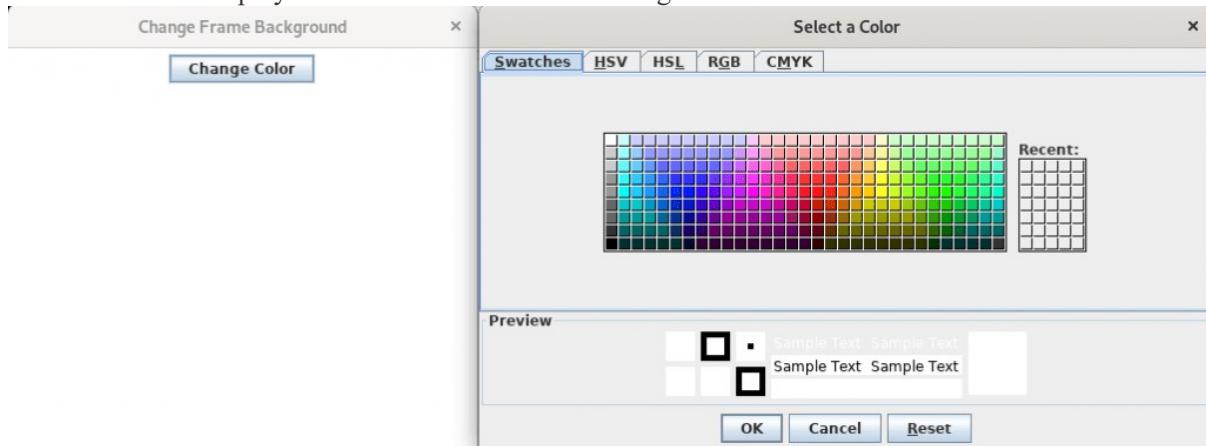
1. To create a color dialog box use **JColorChooser** class.
2. Display the color dialog box using **showDialog(Component,title,initial color)** function.
3. Update the color of the frame

Runtime Test Cases

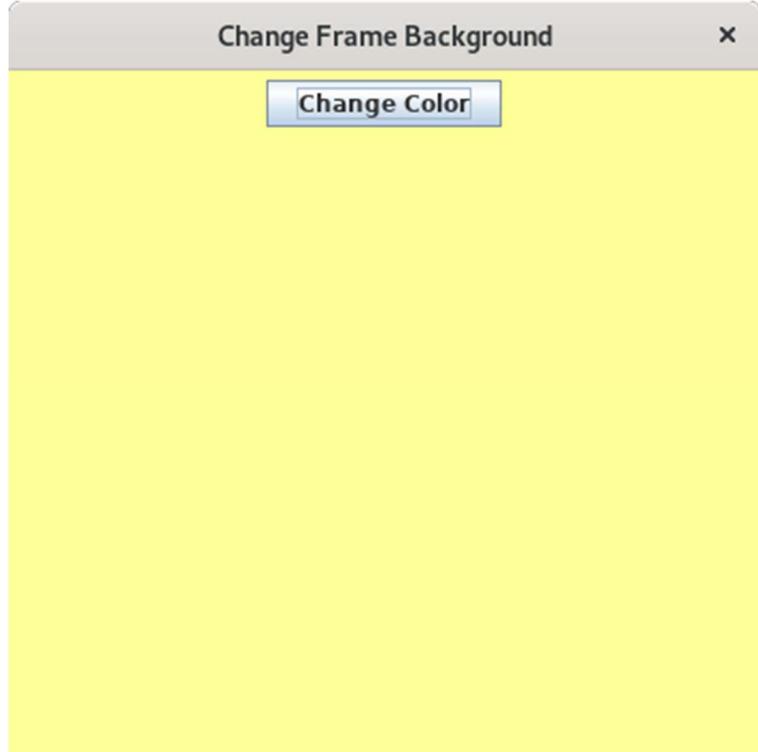
Here's the run time test cases for changing color of frame using color dialog box.

advertisement

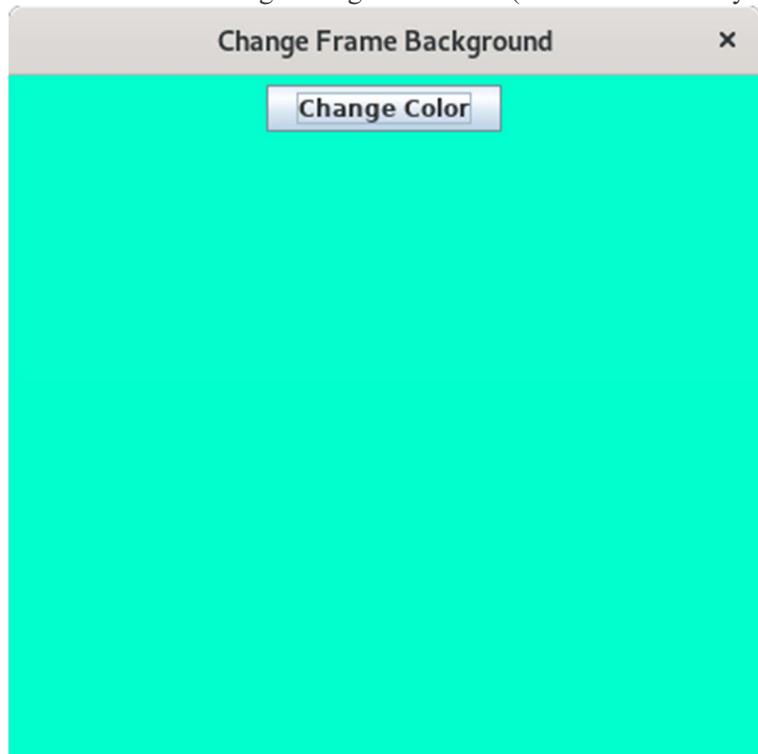
Test case 1 – To display the frame with button and dialog box.



Test case 2 – To change background color. (Selected color – Yellow)



Test case 3 – To change background color. (Selected color – Cyan)



180. Java Program to Solve any Linear Equations

« Prev

Next »

This is java program to solve the system of linear equations. This can be done by first representing equations(vectors) to matrix form, then finding the inverse of the matrix formed by the coefficients of variable and multiplying it with constants.

Here is the source code of the Java Program to Solve any Linear Equation in One Variable. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to solve the linear equations.
2. import java.util.Scanner;
3.
4. public class Solve_Linear_Equation
5. {
6.     public static void main(String args[])
7.     {
8.         char []var = {'x', 'y', 'z', 'w'};
9.         System.out.println("Enter the number of variables in the equations: ");
10.        Scanner input = new Scanner(System.in);
11.        int n = input.nextInt();
12.        System.out.println("Enter the coefficients of each variable for each equations");
13.        System.out.println("ax + by + cz + ... = d");
14.        double [][]mat = new double[n][n];
15.        double [][]constants = new double[n][1];
16.        //input
17.        for(int i=0; i<n; i++)
18.        {
19.            for(int j=0; j<n; j++)
20.            {
21.                mat[i][j] = input.nextDouble();
22.            }
23.            constants[i][0] = input.nextDouble();
24.        }
25.        //Matrix representation
26.        for(int i=0; i<n; i++)
27.        {
28.            for(int j=0; j<n; j++)
29.            {
30.                System.out.print(" "+mat[i][j]);
31.            }
32.            System.out.print(" "+ var[i]);
33.            System.out.print(" = "+ constants[i][0]);
34.            System.out.println();
35.        }
36.        //inverse of matrix mat[][]
37.        double inverted_mat[][] = invert(mat);
38.        System.out.println("The inverse is: ");
39.        for (int i=0; i<n; ++i)
40.        {
41.            for (int j=0; j<n; ++j)
42.            {
43.                System.out.print(inverted_mat[i][j]+" ");
44.            }
45.            System.out.println();
46.        }
47.        //Multiplication of mat inverse and constants
48.        double result[][] = new double[n][1];
49.        for (int i = 0; i < n; i++)
50.        {
51.            for (int j = 0; j < 1; j++)
52.            {
53.                for (int k = 0; k < n; k++)
54.                {
55.                    result[i][j] = result[i][j] + inverted_mat[i][k] * constants[k][j];
```

```

56.         }
57.     }
58. }
59. System.out.println("The product is:");
60. for(int i=0; i<n; i++)
61. {
62.     System.out.println(result[i][0] + " ");
63. }
64. input.close();
65.
66. }
67.
68. public static double[][] invert(double a[][])
69. {
70.     int n = a.length;
71.     double x[][] = new double[n][n];
72.     double b[][] = new double[n][n];
73.     int index[] = new int[n];
74.     for (int i=0; i<n; ++i)
75.         b[i][i] = 1;
76.
77. // Transform the matrix into an upper triangle
78. gaussian(a, index);
79.
80. // Update the matrix b[i][j] with the ratios stored
81. for (int i=0; i<n-1; ++i)
82.     for (int j=i+1; j<n; ++j)
83.         for (int k=0; k<n; ++k)
84.             b[index[j]][k]
85.                 -= a[index[j]][i]*b[index[i]][k];
86.
87. // Perform backward substitutions
88. for (int i=0; i<n; ++i)
89. {
90.     x[n-1][i] = b[index[n-1]][i]/a[index[n-1]][n-1];
91.     for (int j=n-2; j>=0; --j)
92.     {
93.         x[j][i] = b[index[j]][i];
94.         for (int k=j+1; k<n; ++k)
95.         {
96.             x[j][i] -= a[index[j]][k]*x[k][i];
97.         }
98.         x[j][i] /= a[index[j]][j];
99.     }
100. }
101. return x;
102. }
103.
104.// Method to carry out the partial-pivoting Gaussian
105.// elimination. Here index[] stores pivoting order.
106.
107. public static void gaussian(double a[][], int index[])
108. {
109.     int n = index.length;
110.     double c[] = new double[n];
111.
112. // Initialize the index
113.     for (int i=0; i<n; ++i)
114.         index[i] = i;
115.
116. // Find the rescaling factors, one from each row
117.     for (int i=0; i<n; ++i)
118.     {
119.         double c1 = 0;
120.         for (int j=0; j<n; ++j)
121.         {

```

```

122.         double c0 = Math.abs(a[i][j]);
123.         if (c0 > c1) c1 = c0;
124.     }
125.     c[i] = c1;
126. }
127.
128. // Search the pivoting element from each column
129. int k = 0;
130. for (int j=0; j<n-1; ++j)
131. {
132.     double pi1 = 0;
133.     for (int i=j; i<n; ++i)
134.     {
135.         double pi0 = Math.abs(a[index[i]][j]);
136.         pi0 /= c[index[i]];
137.         if (pi0 > pi1)
138.         {
139.             pi1 = pi0;
140.             k = i;
141.         }
142.     }
143.
144. // Interchange rows according to the pivoting order
145. int itmp = index[j];
146. index[j] = index[k];
147. index[k] = itmp;
148. for (int i=j+1; i<n; ++i)
149. {
150.     double pj = a[index[i]][j]/a[index[j]][j];
151.
152. // Record pivoting ratios below the diagonal
153.     a[index[i]][j] = pj;
154.
155. // Modify other elements accordingly
156.     for (int l=j+1; l<n; ++l)
157.         a[index[i]][l] -= pj*a[index[j]][l];
158.     }
159. }
160. }
161. }

```

Output:

advertisement

```

$ javac Solve_Linear_Equation.java
$ java Solve_Linear_Equation
Enter the number of variables in the equations:
2
Enter the coefficients of each variable for each equations
ax + by + cz + ... = d
1 2 3
3 2 1
1.0 2.0 x = 3.0
3.0 2.0 y = 1.0

```

The inverse is:
-0.4999999999999994 0.5
0.749999999999999 -0.2499999999999997

The product is:
-0.9999999999999998
1.999999999999996

181. Java Program to Solve the 0-1 Knapsack Problem

[« Prev](#)

[Next »](#)

This is java program to implement 0/1 Knapsack problem. The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a mass and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Here is the source code of the Java Program to Solve the 0-1 Knapsack Problem. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to implement a 0/1 knapsack algorithm
2. import java.util.Scanner;
3.
4. public class Zero_One_Knapsack
5. {
6.     public void solve(int[] wt, int[] val, int W, int N)
7.     {
8.         int NEGATIVE_INFINITY = Integer.MIN_VALUE;
9.         int[][] m = new int[N + 1][W + 1];
10.        int[][] sol = new int[N + 1][W + 1];
11.        for (int i = 1; i <= N; i++)
12.        {
13.            for (int j = 0; j <= W; j++)
14.            {
15.                int m1 = m[i - 1][j];
16.                int m2 = NEGATIVE_INFINITY;
17.                if (j >= wt[i])
18.                    m2 = m[i - 1][j - wt[i]] + val[i];
19.                m[i][j] = Math.max(m1, m2);
20.                sol[i][j] = m2 > m1 ? 1 : 0;
21.            }
22.        }
23.        int[] selected = new int[N + 1];
24.        for (int n = N, w = W; n > 0; n--)
25.        {
26.            if (sol[n][w] != 0)
27.            {
28.                selected[n] = 1;
29.                w = w - wt[n];
30.            }
31.            else
32.                selected[n] = 0;
33.        }
34.        System.out.print("\nItems with weight ");
35.        for (int i = 1; i < N + 1; i++)
36.            if (selected[i] == 1)
37.                System.out.print(val[i] + " ");
38.        System.out.println("are selected by knapsack algorithm.");
39.    }
40.    public static void main (String[] args)
41.    {
42.        Scanner scan = new Scanner(System.in);
43.        Zero_One_Knapsack ks = new Zero_One_Knapsack();
44.
45.        System.out.println("Enter number of elements ");
46.        int n = scan.nextInt();
47.
48.        int[] wt = new int[n + 1];
49.        int[] val = new int[n + 1];
50.
```

```
51. System.out.println("Enter weight for "+ n +" elements");
52. for (int i = 1; i <= n; i++)
53.     wt[i] = scan.nextInt();
54. System.out.println("Enter value for "+ n +" elements");
55. for (int i = 1; i <= n; i++)
56.     val[i] = scan.nextInt();
57.
58. System.out.println("Enter knapsack weight ");
59. int W = scan.nextInt();
60.
61. ks.solve(wt, val, W, n);
62. scan.close();
63. }
64. }
```

Output:

advertisement

```
$ javac Zero_One_Knapsack.java
$ java Zero_One_Knapsack
```

Enter number of elements

5

Enter weight for 5 elements

01 56 42 78 12

Enter value for 5 elements

50 30 20 10 50

Enter knapsack weight

150

Items with weight 50 30 20 50 are selected by knapsack algorithm.

182. Java Program to Solve Knapsack Problem Using Dynamic Programming

[« Prev](#)

[Next »](#)

This is java program to implement Knapsack problem using Dynamic programming. Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. Consider all subsets of items and calculate the total weight and value of all subsets. Consider the only subsets whose total weight is smaller than W. From all such subsets, pick the maximum value subset.

Here is the source code of the Java Program to Solve Knapsack Problem Using Dynamic Programming. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is the java program to implement the knapsack problem using Dynamic Programming
2. import java.util.Scanner;
3.
4. public class Knapsack_DP
5. {
6.     static int max(int a, int b)
7.     {
8.         return (a > b)? a : b;
9.     }
10.    static int knapSack(int W, int wt[], int val[], int n)
11.    {
12.        int i, w;
13.        int [][]K = new int[n+1][W+1];
14.
15.        // Build table K[][] in bottom up manner
16.        for (i = 0; i <= n; i++)
17.        {
18.            for (w = 0; w <= W; w++)
19.            {
20.                if (i==0 || w==0)
21.                    K[i][w] = 0;
22.                else if (wt[i-1] <= w)
23.                    K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
24.                else
25.                    K[i][w] = K[i-1][w];
26.            }
27.        }
28.
29.        return K[n][W];
30.    }
31.
32.    public static void main(String args[])
33.    {
34.        Scanner sc = new Scanner(System.in);
35.        System.out.println("Enter the number of items: ");
36.        int n = sc.nextInt();
37.        System.out.println("Enter the items weights: ");
38.        int []wt = new int[n];
39.        for(int i=0; i<n; i++)
40.            wt[i] = sc.nextInt();
41.
42.        System.out.println("Enter the items values: ");
43.        int []val = new int[n];
44.        for(int i=0; i<n; i++)
45.            val[i] = sc.nextInt();
46.
47.        System.out.println("Enter the maximum capacity: ");
48.        int W = sc.nextInt();
49.
50.        System.out.println("The maximum value that can be put in a knapsack of capacity W is: " + knapSack(W, wt, val,
51. n));
52.        sc.close();
```

```
52. }  
53. }
```

Output:

advertisement

```
$ javac Knapsack_DP.java  
$ java Knapsack_DP  
  
Enter the number of items:  
5  
Enter the items weights:  
01 56 42 78 12  
Enter the items values:  
50 30 20 10 50  
Enter the maximum capacity:  
150  
The maximum value that can be put in a knapsack of capacity W is: 150
```

183. Java Program to Generate Random Numbers Using Probability Distribution Function

[« Prev](#)

[Next »](#)

This is a java program to generate random numbers using a probability distribution. Probability distribution is based on probability density function. A probability density function (pdf), or density of a continuous random variable, is a function that describes the relative likelihood for this random variable to take on a given value. The probability of the random variable falling within a particular range of values is given by the integral of this variable's density over that range—that is, it is given by the area under the density function but above the horizontal axis and between the lowest and greatest values of the range.

Here is the source code of the Java Program to Generate Random Numbers Using Probability Distribution Function. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to generate a random numbers based on probability desity function of spinner
2. //pdf(x) = 1 if x>360
3. //      = 0 if x<0
4. //      = x/360 otherwise
5. import java.util.Random;
6.
7. public class Probability_distribution_Function_Random_Numbers
8. {
9.     static int N = 10;
10.    public static void main(String args[])
11.    {
12.        Random random = new Random();
13.        int p=0;
14.        for(int i=0; i<N; i++)
15.        {
16.            p = random.nextInt(400);
17.            if(p > 360)
18.                System.out.println(1 + " ");
19.            else if(p < 0)
20.                System.out.println(0 + " ");
21.            else
22.                System.out.println(p*0.1/360 + " ");
23.        }
24.    }
25. }
```

Output:

advertisement

```
$ javac Probability_distribution_Function_Random_Numbers.java
$ java Probability_distribution_Function_Random_Numbers
The random numbers are:
0.0852777777777779
0.0736111111111111
0.0072222222222223
0.0869444444444445
1
1
0.0552777777777779
0.0952777777777778
0.0488888888888889
0.0169444444444446
```

184. Java Program to Use rand and srand Functions

[« Prev](#)

[Next »](#)

This is a java program to generate random numbers using rand() and srand() functions. Java provides Random class that generates a random numbers. rand() gives long random numbers. srand() provides unique numbers.

Here is the source code of the Java Program to Use rand and srand Functions. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to generate a random numbers using rand() and srand()
2. //rand() gives long random numbers
3. //srand() provides unique numbers
4. import java.util.Random;
5. import java.util.UUID;
6.
7. public class Rand_and_Srand
8. {
9.     public static void main(String args[])
10.    {
11.        System.out.println("The numbers using rand");
12.        for(int i=0; i<5; i++)
13.        {
14.            Random rand = new Random();
15.            System.out.println(Math.abs(rand.nextInt()));
16.        }
17.
18.        System.out.println("The numbers using srand");
19.        for(int i=0; i<5; i++)
20.        {
21.            System.out.println(Math.abs(UUID.randomUUID().getMostSignificantBits()));
22.        }
23.    }
24. }
```

Output:

advertisement

```
$ javac Rand_and_Srand.java
$ java Rand_and_Srand
```

The numbers using rand

1339557437

636169175

1207287888

1539694038

1040189301

The numbers using srand

301709257092546335

8798470719933102847

3480203219570178904

3272351410737399038

2158529096808811162

185. Java Program to Solve the Fractional Knapsack Problem

[« Prev](#)

[Next »](#)

This is a java program to implement a standard fractional knapsack problem. It is an algorithmic problem in combinatorial optimization in which the goal is to fill a container (the “knapsack”) with fractional amounts of different materials chosen to maximize the value of the selected materials.

Here is the source code of the Java Program to Solve the Fractional Knapsack Problem. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to implement a fractional knapsack problem
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class Fractional_Knapsack
6. {
7.     public static void main(String args[]) throws IOException
8.     {
9.         int i,j=0,max_qty,m,n;
10.        float sum=0,max;
11.        Scanner sc = new Scanner(System.in);
12.        int array[][]=new int[2][20];
13.        System.out.println("Enter no of items");
14.        n=sc.nextInt();
15.
16.        System.out.println("Enter the weights of each items");
17.        for(i=0;i<n;i++)
18.            array[0][i]=sc.nextInt();
19.
20.        System.out.println("Enter the values of each items");
21.        for(i=0;i<n;i++)
22.            array[1][i]=sc.nextInt();
23.
24.        System.out.println("Enter maximum volume of knapsack :");
25.        max_qty=sc.nextInt();
26.
27.        m=max_qty;
28.        while(m>=0)
29.        {
30.            max=0;
31.            for(i=0;i<n;i++)
32.            {
33.                if(((float)array[1][i])/((float)array[0][i])>max)
34.                {
35.                    max=((float)array[1][i])/((float)array[0][i]);
36.                    j=i;
37.                }
38.            }
39.            if(array[0][j]>m)
40.            {
41.                System.out.println("Quantity of item number: " + (j+1) + " added is " +m);
42.                sum+=m*max;
43.                m=-1;
44.            }
45.            else
46.            {
47.                System.out.println("Quantity of item number: " + (j+1) + " added is " + array[0][j]);
48.                m-=array[0][j];
49.                sum+=(float)array[1][j];
50.                array[1][j]=0;
51.            }
52.        }
53.        System.out.println("The total profit is " + sum);
```

```
54.     sc.close();
55. }
56.}
```

Output:

advertisement

```
$ javac Fractional_Knapsack.java
$ java Fractional_Knapsack
```

Enter no of items

5

Enter the weights of each items

10 20 30 40 50

Enter the values of each items

5 4 3 2 1

Enter maximum volume of knapsack :

80

Quantity of item number: 1 added is 10

Quantity of item number: 2 added is 20

Quantity of item number: 3 added is 30

Quantity of item number: 4 added is 20

The total profit is 13.0

186. Java Program to Perform LU Decomposition of any Matrix

[« Prev](#)

[Next »](#)

This is a java program to LU Decomposition of a given matrix. LU decomposition is the process of reducing single matrix into 2 matrices such that, upon multiplication we get the original matrix, having property that one of them is lower triangular matrix and other one is upper triangular matrix.

Here is the source code of the Java Program to Perform LU Decomposition of any Matrix. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to calculate the LU decomposition of the given matrix
2. import java.util.Scanner;
3.
4. public class LUDecomposition
5. {
6.     public static void main(String args[])
7.     {
8.         System.out.println("Enter the dimension of the matrix:");
9.         Scanner sc = new Scanner(System.in);
10.        int n = sc.nextInt();
11.        double [][]mat = new double[n][n];
12.        for(int i=0; i<n; i++)
13.            for(int j=0; j<n; j++)
14.                mat[i][j] = sc.nextDouble();
15.
16.        if(n==2)
17.        {
18.            double [][]l = new double[n][n];
19.            l[0][0] = l[1][1] = 1;
20.            l[0][1] = 0;
21.
22.            double [][]u = new double[n][n];
23.            u[1][0] = 0;
24.
25.            u[0][0] = mat[0][0];
26.            u[0][1] = mat[0][1];
27.
28.            l[1][0] = mat[1][0]/mat[0][0];
29.            u[1][1] = mat[1][1] - (l[1][0]*u[0][1]); //mat[2][2]-(mat[2][1]*mat[1][2]/mat[1][1]);
30.
31.            System.out.println("The L Component is:");
32.            for(int i=0; i<n; i++)
33.            {
34.                for(int j=0; j<n; j++)
35.                    System.out.print(" "+l[i][j]);
36.                System.out.println();
37.            }
38.            System.out.println("The U Component is:");
39.            for(int i=0; i<n; i++)
40.            {
41.                for(int j=0; j<n; j++)
42.                    System.out.print(" "+u[i][j]);
43.                System.out.println();
44.            }
45.
46.        }
47.        if(n==3)
48.        {
49.            double [][]l = new double[n][n];
50.            l[0][0] = l[1][1] = l[2][2] = 1;
51.            l[0][1] = l[0][2] = l[1][2] = 0;
52.
53.            double [][]u = new double[n][n];
```

```

54.     u[1][0] = u[2][0] = u[2][1] = 0;
55.
56.     u[0][0] = mat[0][0];
57.     u[0][1] = mat[0][1];
58.     u[0][2] = mat[0][2];
59.
60.     l[1][0] = mat[1][0]/mat[0][0];
61.     u[1][1] = mat[1][1] - (l[1][0]*u[0][1]); //mat[2][2]-(mat[2][1]*mat[1][2]/mat[1][1]);
62.     u[1][2] = mat[1][2] - (l[1][0]*u[0][2]);
63.
64.     l[2][0] = mat[2][0]/u[0][0];
65.     l[2][1] = (mat[2][1] - l[2][0]*u[0][1])/u[1][1];
66.     u[2][2] = mat[2][2] - (l[2][0]*u[0][2]) - (l[2][1]*u[1][2]);
67.
68.     System.out.println("The L Component is:");
69.     for(int i=0; i<n; i++)
70.     {
71.         for(int j=0; j<n; j++)
72.             System.out.print(" "+l[i][j]);
73.         System.out.println();
74.     }
75.     System.out.println("The U Component is:");
76.     for(int i=0; i<n; i++)
77.     {
78.         for(int j=0; j<n; j++)
79.             System.out.print(" "+u[i][j]);
80.         System.out.println();
81.     }
82. }
83. sc.close();
84. }
85.

```

Output:

advertisement

```

$ javac LUDecomposition.java
$ java LUDecomposition.java
Enter the dimension of the matrix:
3
2 3 1
4 5 1
1 1 1
The L Component is:
1.0 0.0 0.0
2.0 1.0 0.0
0.5 -1.0 1.0
The U Component is:
2.0 3.0 1.0
0.0 -1.0 -1.0
0.0 0.0 -0.5

```

187. Java Program to Compute Discrete Fourier Transform Using the Fast Fourier Transform Approach

« Prev

Next »

This is the java implementation of performing Discrete Fourier Transform using Fast Fourier Transform algorithm. This class finds the DFT of N (power of 2) complex elements, generated randomly, using FFT. Further verification is done by taking the Inverse Discrete Fourier Transform, again using FFT.

Here is the source code of the Java Program to Compute Discrete Fourier Transform Using the Fast Fourier Transform Approach. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. // This is a sample program to perform DFT using FFT, FFT is performed on random input sequence
2. public class FFT
3. {
4.     public static class Complex
5.     {
6.         private final double re; // the real part
7.         private final double im; // the imaginary part
8.
9.         // create a new object with the given real and imaginary parts
10.        public Complex(double real, double imag)
11.        {
12.            re = real;
13.            im = imag;
14.        }
15.
16.        // return a string representation of the invoking Complex object
17.        public String toString()
18.        {
19.            if (im == 0)
20.                return re + "";
21.            if (re == 0)
22.                return im + "i";
23.            if (im < 0)
24.                return re + " - " + (-im) + "i";
25.            return re + " + " + im + "i";
26.        }
27.
28.        // return abs/modulus/magnitude and angle/phase/argument
29.        public double abs()
30.        {
31.            return Math.hypot(re, im);
32.        } // Math.sqrt(re*re + im*im)
33.
34.        public double phase()
35.        {
36.            return Math.atan2(im, re);
37.        } // between -pi and pi
38.
39.        // return a new Complex object whose value is (this + b)
40.        public Complex plus(Complex b)
41.        {
42.            Complex a = this; // invoking object
43.            double real = a.re + b.re;
44.            double imag = a.im + b.im;
45.            return new Complex(real, imag);
46.        }
47.
48.        // return a new Complex object whose value is (this - b)
49.        public Complex minus(Complex b)
```

```

50. {
51.     Complex a = this;
52.     double real = a.re - b.re;
53.     double imag = a.im - b.im;
54.     return new Complex(real, imag);
55. }
56.
57. // return a new Complex object whose value is (this * b)
58. public Complex times(Complex b)
59. {
60.     Complex a = this;
61.     double real = a.re * b.re - a.im * b.im;
62.     double imag = a.re * b.im + a.im * b.re;
63.     return new Complex(real, imag);
64. }
65.
66. // scalar multiplication
67. // return a new object whose value is (this * alpha)
68. public Complex times(double alpha)
69. {
70.     return new Complex(alpha * re, alpha * im);
71. }
72.
73. // return a new Complex object whose value is the conjugate of this
74. public Complex conjugate()
75. {
76.     return new Complex(re, -im);
77. }
78.
79. // return a new Complex object whose value is the reciprocal of this
80. public Complex reciprocal()
81. {
82.     double scale = re * re + im * im;
83.     return new Complex(re / scale, -im / scale);
84. }
85.
86. // return the real or imaginary part
87. public double re()
88. {
89.     return re;
90. }
91. public double im()
92. {
93.     return im;
94. }
95.
96. // return a / b
97. public Complex divides(Complex b)
98. {
99.     Complex a = this;
100.    return a.times(b.reciprocal());
101. }
102.
103. // return a new Complex object whose value is the complex exponential of
104. // this
105. public Complex exp()
106. {
107.     return new Complex(Math.exp(re) * Math.cos(im), Math.exp(re)
108.                     * Math.sin(im));
109. }
110.
111. // return a new Complex object whose value is the complex sine of this
112. public Complex sin()
113. {
114.     return new Complex(Math.sin(re) * Math.cosh(im), Math.cos(re)
115.                     * Math.sinh(im));

```

```

116.     }
117.
118.    // return a new Complex object whose value is the complex cosine of this
119.    public Complex cos()
120.    {
121.        return new Complex(Math.cos(re) * Math.cosh(im), -Math.sin(re)
122.                      * Math.sinh(im));
123.    }
124.
125.    // return a new Complex object whose value is the complex tangent of
126.    // this
127.    public Complex tan()
128.    {
129.        return sin().divides(cos());
130.    }
131.
132.    // a static version of plus
133.    public static Complex plus(Complex a, Complex b)
134.    {
135.        double real = a.re + b.re;
136.        double imag = a.im + b.im;
137.        Complex sum = new Complex(real, imag);
138.        return sum;
139.    }
140.
141.    // compute the FFT of x[], assuming its length is a power of 2
142.    public static Complex[] fft(Complex[] x)
143.    {
144.        int N = x.length;
145.
146.        // base case
147.        if (N == 1)
148.            return new Complex[] { x[0] };
149.
150.        // radix 2 Cooley-Tukey FFT
151.        if (N % 2 != 0)
152.        {
153.            throw new RuntimeException("N is not a power of 2");
154.        }
155.
156.        // fft of even terms
157.        Complex[] even = new Complex[N / 2];
158.        for (int k = 0; k < N / 2; k++)
159.        {
160.            even[k] = x[2 * k];
161.        }
162.        Complex[] q = fft(even);
163.
164.        // fft of odd terms
165.        Complex[] odd = even; // reuse the array
166.        for (int k = 0; k < N / 2; k++)
167.        {
168.            odd[k] = x[2 * k + 1];
169.        }
170.        Complex[] r = fft(odd);
171.
172.        // combine
173.        Complex[] y = new Complex[N];
174.        for (int k = 0; k < N / 2; k++)
175.        {
176.            double kth = -2 * k * Math.PI / N;
177.            Complex wk = new Complex(Math.cos(kth), Math.sin(kth));
178.            y[k] = q[k].plus(wk.times(r[k]));
179.            y[k + N / 2] = q[k].minus(wk.times(r[k]));
180.        }
181.        return y;

```

```

182.     }
183.
184.     // compute the inverse FFT of x[], assuming its length is a power of 2
185.     public static Complex[] ifft(Complex[] x)
186.     {
187.         int N = x.length;
188.         Complex[] y = new Complex[N];
189.
190.         // take conjugate
191.         for (int i = 0; i < N; i++)
192.         {
193.             y[i] = x[i].conjugate();
194.         }
195.
196.         // compute forward FFT
197.         y = fft(y);
198.
199.         // take conjugate again
200.         for (int i = 0; i < N; i++)
201.         {
202.             y[i] = y[i].conjugate();
203.         }
204.
205.         // divide by N
206.         for (int i = 0; i < N; i++)
207.         {
208.             y[i] = y[i].times(1.0 / N);
209.         }
210.
211.         return y;
212.
213.     }
214.
215.     // display an array of Complex numbers to standard output
216.     public static void show(Complex[] x, String title)
217.     {
218.         System.out.println(title);
219.         for (int i = 0; i < x.length; i++)
220.         {
221.             System.out.println(x[i]);
222.         }
223.         System.out.println();
224.     }
225.
226.     public static void main(String[] args)
227.     {
228.         int N = 8;//Integer.parseInt(args[0]);
229.         Complex[] x = new Complex[N];
230.
231.         // original data
232.         for (int i = 0; i < N; i++)
233.         {
234.             x[i] = new Complex(i, 0);
235.             x[i] = new Complex(-2 * Math.random() + 1, 0);
236.         }
237.         show(x, "x");
238.
239.         // FFT of original data
240.         Complex[] y = fft(x);
241.         show(y, "y = fft(x)");
242.
243.         // take inverse FFT
244.         Complex[] z = ifft(y);
245.         show(z, "z = ifft(y)");
246.
247.     }

```

```
248.  
249. }  
250. }
```

Output:

advertisement

```
$ javac FFT.java  
$ java FFT  
  
x  
0.5568836254037923  
0.8735842104393365  
0.6099699812709252  
0.5631502515566189  
-0.518857260970139  
-0.5946393148293805  
0.47144753318047794  
-0.3501597962417593  
  
y = fft(x)  
1.6113792298098721  
1.4681239692650163 - 1.8225209872296184i  
-1.0433911500177497 - 0.06595444029509645i  
0.6833578034828462 - 1.545476091048724i  
0.6275085279602408  
0.6833578034828462 + 1.545476091048724i  
-1.0433911500177497 + 0.06595444029509645i  
1.4681239692650163 + 1.8225209872296184i  
  
z = ifft(y)  
0.5568836254037923  
0.8735842104393365 - 5.652078740871965E-17i  
0.6099699812709252 - 4.24102681660054E-18i  
0.5631502515566189 - 5.4501515053796015E-17i  
-0.518857260970139  
-0.5946393148293805 + 5.4501515053796015E-17i  
0.47144753318047794 + 4.24102681660054E-18i  
-0.3501597962417593 + 5.652078740871965E-17i
```

188. Java Perform to a 2D FFT Inplace Given a Complex 2D Array

[« Prev](#)

[Next »](#)

This is the java implementation of performing Discrete Fourier Transform using Fast Fourier Transform algorithm. This class finds the DFT of N (power of 2) complex elements, generated randomly, using FFT. The input to the class is a two dimensional array of sequence.

Here is the source code of the Java Perform to a 2D FFT Inplace Given a Complex 2D Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to perform 2D FFT inplace
2. import java.util.Scanner;
3.
4. public class TwoD_FFT
5. {
6.     static void twoDfft(double[][] inputData, double[][] realOut,
7.                         double[][] imagOut, double[][] amplitudeOut)
8.     {
9.         int height = inputData.length;
10.        int width = inputData[0].length;
11.
12.        // Two outer loops iterate on output data.
13.        for (int yWave = 0; yWave < height; yWave++)
14.        {
15.            for (int xWave = 0; xWave < width; xWave++)
16.            {
17.                // Two inner loops iterate on input data.
18.                for (int ySpace = 0; ySpace < height; ySpace++)
19.                {
20.                    for (int xSpace = 0; xSpace < width; xSpace++)
21.                    {
22.                        // Compute real, imag, and amplitude.
23.                        realOut[yWave][xWave] += (inputData[ySpace][xSpace] * Math
24.                            .cos(2
25.                                * Math.PI
26.                                * ((1.0 * xWave * xSpace / width) + (1.0
27.                                    * yWave * ySpace / height)))
28.                            / Math.sqrt(width * height));
29.                        imagOut[yWave][xWave] -= (inputData[ySpace][xSpace] * Math
30.                            .sin(2
31.                                * Math.PI
32.                                * ((1.0 * xWave * xSpace / width) + (1.0
33.                                    * yWave * ySpace / height)))
34.                            / Math.sqrt(width * height));
35.                        amplitudeOut[yWave][xWave] = Math
36.                            .sqrt(realOut[yWave][xWave]
37.                                * realOut[yWave][xWave]
38.                                + imagOut[yWave][xWave]
39.                                * imagOut[yWave][xWave]);
40.                    }
41.                    System.out.println(realOut[yWave][xWave] + " +
42.                        + imagOut[yWave][xWave] + " i");
43.                }
44.            }
45.        }
46.    }
47.
48.    public static void main(String args[])
49.    {
50.        System.out.println("Enter the size: ");
51.        Scanner sc = new Scanner(System.in);
52.        int n = sc.nextInt();
53.        double[][] input = new double[n][n];
```

```
54. double[][] real = new double[n][n];
55. double[][] img = new double[n][n];
56. double[][] amplitude = new double[n][n];
57. System.out.println("Enter the 2D elements ");
58. for (int i = 0; i < n; i++)
59.     for (int j = 0; j < n; j++)
60.         input[i][j] = sc.nextDouble();
61.
62. twoDfft(input, real, img, amplitude);
63.
64. sc.close();
65. }
66. }
```

Output:

advertisement

```
$ javac TwoD_FFT.java
$ java TwoD_FFT
```

Enter the size:

2

Enter the 2D elements

2 3

4 2

```
2.5 + 0.0 i
5.5 + 0.0 i
-0.5 + -1.8369701987210297E-16 i
0.5 + -3.0616169978683826E-16 i
2.5 + 0.0 i
-0.5 + -3.6739403974420594E-16 i
-0.5 + -1.8369701987210297E-16 i
-1.5 + -1.8369701987210297E-16 i
```

189. Java Program to Generate Random Numbers Using Multiply with Carry Method

[« Prev](#)

[Next »](#)

This is the java implementation of Multiply With Carry (MWC) algorithm to generate a set of random numbers. The main advantages of the MWC method are that it invokes simple computer integer arithmetic and leads to very fast generation of sequences of random numbers with immense periods. The formula it uses is, $x(n) = (a*x(n-r) + c(n-1)) \text{mod } n$ and $c(n) = (a*x(n-r) + c(n-1))/n$, where a is any multiplier, m is modulus, c is carry and r is initial number of seeds.

Here is the source code of the Java Program to Generate Random Numbers Using Multiply with Carry Method. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to generate a random numbers based on Multiply with carry method
2. import java.util.Random;
3.
4. public class Multiply_With_Carry
5. {
6.     public static void main(String args[])
7.     {
8.         int max_Sequence_Elements = 10;
9.         Random random = new Random();
10.        int base_b = 2000;
11.        int multiplier_a = random.nextInt(base_b);
12.        int r = 1;
13.        int []c = new int[max_Sequence_Elements];
14.        int []x = new int[max_Sequence_Elements];
15.
16.        c[0] = random.nextInt(multiplier_a);
17.        x[0] = random.nextInt(base_b);
18.
19.        System.out.print("The random number sequence is: " + x[0]);
20.        //generating sequence
21.        for(int i=1; i<max_Sequence_Elements; i++)
22.        {
23.            x[i] = (multiplier_a*x[i-r] + c[i-1]) % base_b;
24.            c[i] = (multiplier_a*x[i-r] + c[i-1]) / base_b;
25.            System.out.print(" " + x[i]);
26.        }
27.    }
28.}
```

Output:

advertisement

```
$ javac Multiply_With_Carry.java
$ java Multiply_With_Carry
The random number sequence is: 795 382 1487 260 475 1798 1347 722 1389 63
```

190. Java Program to Perform Partition of an Integer in All Possible Ways

[« Prev](#)

[Next »](#)

This is a java program to perform partition of an integer in all possible ways. Every partition when added should result in the given integer.

Here is the source code of the Java Program to Perform Partition of an Integer in All Possible Ways. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is sample program to print a unique partitions of a given number
2. import java.util.Scanner;
3.
4. public class Unique_Partitions_Number
5. {
6.     public static void print(int[]p, int n)
7.     {
8.         for(int i=0; i<n; i++)
9.             System.out.print(p[i]+" ");
10.        System.out.println();
11.    }
12.    public static void generateUniquePartition(int n)
13.    {
14.        int []p = new int[n+n];
15.        int k = 0;
16.        p[k] = n;
17.        while(true)
18.        {
19.            print(p, k+1);
20.            int rem_value = 0;
21.            while(k >= 0 && p[k] == 1)
22.            {
23.                rem_value += p[k];
24.                k--;
25.            }
26.            if(k < 0)
27.                return;
28.
29.            p[k]--;
30.            rem_value++;
31.
32.            while(rem_value > p[k])
33.            {
34.                p[k+1] = p[k];
35.                rem_value -= p[k];
36.                k++;
37.            }
38.            p[k+1] = rem_value;
39.            k++;
40.        }
41.    }
42.    public static void main(String args[])
43.    {
44.        System.out.println("Unique Partitioning of a given number");
45.        System.out.println("Enter the number:");
46.        Scanner sc = new Scanner(System.in);
47.        int n = sc.nextInt();
48.        generateUniquePartition(n);
49.        sc.close();
50.    }
51.}
```

Output:

advertisement

```
$ javac Unique_Partitions_Number.java
$ java Unique_Partitions_Number
Unique Partitioning of a given number
Enter the number:
4

4
3 1
2 2
2 1 1
1 1 1 1
```

191. Java Program to Implement Naor-Reingold Pseudo Random Function

[« Prev](#)

[Next »](#)

This is a java program generate pseudo-random numbers using Naor-Reingold Pseudo-Random function. Let p and l be prime numbers with $l \mid p-1$. Select an element g in F_p of multiplicative order l. Then for each n-dimensional vector $a = (a_1, \dots, a_n)$. $F_a(x) = g^{(a_1 \cdot x_1 * a_2 \cdot x_2 \dots)}$.

Here is the source code of the Java Program to Implement Naor-Reingold Pseudo Random Function. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate a random numbers using Naor-Reingold Psedurandom Function
2. import java.util.Random;
3.
4. public class Naor_Reingold
5. {
6.     public static void main(String args[])
7.     {
8.         int p=7, l=3, g=2, n=4, x;
9.         int []a = {1,2,2,1};
10.        int []bin = new int[4];
11.        Random random = new Random();
12.        System.out.println("The Random numbers are: ");
13.        for(int i=0; i<10; i++)
14.        {
15.            x = random.nextInt(17);
16.            for(int j=3; j>=0; j--)
17.            {
18.                bin[j] = x%2;
19.                x/=2;
20.            }
21.            int mul = 1;
22.            for(int k=0; k<4; k++)
23.                mul *= Math.pow(a[k], bin[k]);
24.            System.out.println(Math.pow(g, mul));
25.        }
26.    }
27. }
```

Output:

advertisement

```
$ javac Naor_Reingold.java
$ java Naor_Reingold
The Random numbers are:
2.0
4.0
2.0
2.0
2.0
16.0
4.0
16.0
16.0
4.0
```

192. Java Program to Find Path Between Two Nodes in a Graph

[« Prev](#)

[Next »](#)

This is a java program find a path between two nodes in a graph if it exists. Path exists between two nodes if there is a connectivity between them through other nodes. A simple run of Breadth First Search will decide whether there is path between two given nodes or not.

Here is the source code of the Java Program to Find Path Between Two Nodes in a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to find the minimum wire length between two component in a electrical circuits
2. import java.util.*;
3. class Node
4. {
5.     public int label; // this node's label (parent node in path tree)
6.     public int weight; // weight of edge to this node (distance to start)
7.
8.     public Node(int v, int w)
9.     {
10.         label = v;
11.         weight = w;
12.     }
13. }
14.
15. public class ShortestPath
16. {
17.     public static Scanner in; //for standard input
18.     public static int n, m; // n = #vertices, m = #edges
19.     public static LinkedList[] graph; // adjacency list representation
20.     public static int start, end; // start and end points for shortest path
21.
22.     public static void main(String[] args)
23.     {
24.         in = new Scanner(System.in);
25.
26.         // Input the graph:
27.         System.out
28.             .println("Enter the number of components and wires in a circuit:");
29.         n = in.nextInt();
30.         m = in.nextInt();
31.
32.         // Initialize adjacency list structure to empty lists:
33.         graph = new LinkedList[n];
34.         for (int i = 0; i < n; i++)
35.             graph[i] = new LinkedList();
36.
37.         // Add each edge twice, once for each endpoint:
38.         System.out
39.             .println("Mention the wire between components and its length:");
40.         for (int i = 0; i < m; i++)
41.         {
42.             int v1 = in.nextInt();
43.             int v2 = in.nextInt();
44.             int w = in.nextInt();
45.             graph[v1].add(new Node(v2, w));
46.             graph[v2].add(new Node(v1, w));
47.         }
48.
49.         // Input starting and ending vertices:
50.         System.out
51.             .println("Enter the start and end for which length is to be minimized: ");
52.         start = in.nextInt();
53.         end = in.nextInt();
```

```

54.
55. //FOR DEBUGGING ONLY:
56. displayGraph();
57.
58. //Print shortest path from start to end:
59. shortest();
60. }
61.
62. public static void shortest()
63. {
64.     boolean[] done = new boolean[n];
65.     Node[] table = new Node[n];
66.     for (int i = 0; i < n; i++)
67.         table[i] = new Node(-1, Integer.MAX_VALUE);
68.
69.     table[start].weight = 0;
70.
71.     for (int count = 0; count < n; count++)
72.     {
73.         int min = Integer.MAX_VALUE;
74.         int minNode = -1;
75.         for (int i = 0; i < n; i++)
76.             if (!done[i] && table[i].weight < min)
77.             {
78.                 min = table[i].weight;
79.                 minNode = i;
80.             }
81.
82.         done[minNode] = true;
83.
84.         ListIterator iter = graph[minNode].listIterator();
85.         while (iter.hasNext())
86.         {
87.             Node nd = (Node) iter.next();
88.             int v = nd.label;
89.             int w = nd.weight;
90.
91.             if (!done[v] && table[minNode].weight + w < table[v].weight)
92.             {
93.                 table[v].weight = table[minNode].weight + w;
94.                 table[v].label = minNode;
95.             }
96.         }
97.     }
98.     for (int i = 0; i < n; i++)
99.     {
100.         if (table[i].weight < Integer.MAX_VALUE)
101.         {
102.             System.out.print("Wire from " + i + " to " + start
103.                             + " with length " + table[i].weight + ": ");
104.             int next = table[i].label;
105.             while (next >= 0)
106.             {
107.                 System.out.print(next + " ");
108.                 next = table[next].label;
109.             }
110.             System.out.println();
111.         } else
112.             System.out.println("No wire from " + i + " to " + start);
113.     }
114. }
115.
116. public static void displayGraph()
117. {
118.     for (int i = 0; i < n; i++)
119.     {

```

```
120.     System.out.print(i + ": ");
121.     ListIterator nbrs = graph[i].listIterator(0);
122.     while (nbrs.hasNext())
123.     {
124.         Node nd = (Node) nbrs.next();
125.         System.out.print(nd.label + "(" + nd.weight + ") ");
126.     }
127.     System.out.println();
128. }
129. }
130. }
```

Output:

advertisement

```
$ javac ShortestPath.java
$ java ShortestPath
Enter the number of components and wires in a circuit:
4 3
Mention the wire between components and its length:
0 1 2
1 3 3
1 2 2
Enter the start and end for which length is to be minimized:
0 1
0: 1(2)
1: 0(2) 3(3) 2(2)
2: 1(2)
3: 1(3)
Wire from 0 to 0 with length 0:
Wire from 1 to 0 with length 2: 0
Wire from 2 to 0 with length 4: 1 0
Wire from 3 to 0 with length 5: 1 0
```

193. Java Program to Optimize Wire Length in Electrical Circuit

[« Prev](#)

[Next »](#)

This is a java program to find the optimized wire length between any two components in an electric circuits. This problem is similar to the problem of finding the shortest path between any two cities in the state. A simple Dijkstra's algorithm would result in providing the shortest wire length between any two components in electric circuits.

Here is the source code of the Java Program to Optimize Wire Length in Electrical Circuit. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to find the minimum wire length between two component in a electrical circuits
2. import java.util.*;
3. class Node
4. {
5.     public int label; // this node's label (parent node in path tree)
6.     public int weight; // weight of edge to this node (distance to start)
7.
8.     public Node(int v, int w)
9.     {
10.         label = v;
11.         weight = w;
12.     }
13. }
14.
15. public class ShortestPath
16. {
17.     public static Scanner in; //for standard input
18.     public static int n, m; // n = #vertices, m = #edges
19.     public static LinkedList[] graph; // adjacency list representation
20.     public static int start, end; // start and end points for shortest path
21.
22.     public static void main(String[] args)
23.     {
24.         in = new Scanner(System.in);
25.
26.         // Input the graph:
27.         System.out
28.             .println("Enter the number of components and wires in a circuit:");
29.         n = in.nextInt();
30.         m = in.nextInt();
31.
32.         // Initialize adjacency list structure to empty lists:
33.         graph = new LinkedList[n];
34.         for (int i = 0; i < n; i++)
35.             graph[i] = new LinkedList();
36.
37.         // Add each edge twice, once for each endpoint:
38.         System.out
39.             .println("Mention the wire between components and its length:");
40.         for (int i = 0; i < m; i++)
41.         {
42.             int v1 = in.nextInt();
43.             int v2 = in.nextInt();
44.             int w = in.nextInt();
45.             graph[v1].add(new Node(v2, w));
46.             graph[v2].add(new Node(v1, w));
47.         }
48.
49.         // Input starting and ending vertices:
50.         System.out
51.             .println("Enter the start and end for which length is to be minimized: ");
52.         start = in.nextInt();
53.         end = in.nextInt();
```

```

54.
55. //FOR DEBUGGING ONLY:
56. displayGraph();
57.
58. //Print shortest path from start to end:
59. shortest();
60. }
61.
62. public static void shortest()
63. {
64.     boolean[] done = new boolean[n];
65.     Node[] table = new Node[n];
66.     for (int i = 0; i < n; i++)
67.         table[i] = new Node(-1, Integer.MAX_VALUE);
68.
69.     table[start].weight = 0;
70.
71.     for (int count = 0; count < n; count++)
72.     {
73.         int min = Integer.MAX_VALUE;
74.         int minNode = -1;
75.         for (int i = 0; i < n; i++)
76.             if (!done[i] && table[i].weight < min)
77.             {
78.                 min = table[i].weight;
79.                 minNode = i;
80.             }
81.
82.         done[minNode] = true;
83.
84.         ListIterator iter = graph[minNode].listIterator();
85.         while (iter.hasNext())
86.         {
87.             Node nd = (Node) iter.next();
88.             int v = nd.label;
89.             int w = nd.weight;
90.
91.             if (!done[v] && table[minNode].weight + w < table[v].weight)
92.             {
93.                 table[v].weight = table[minNode].weight + w;
94.                 table[v].label = minNode;
95.             }
96.         }
97.     }
98.     for (int i = 0; i < n; i++)
99.     {
100.         if (table[i].weight < Integer.MAX_VALUE)
101.         {
102.             System.out.print("Wire from " + i + " to " + start
103.                             + " with length " + table[i].weight + ": ");
104.             int next = table[i].label;
105.             while (next >= 0)
106.             {
107.                 System.out.print(next + " ");
108.                 next = table[next].label;
109.             }
110.             System.out.println();
111.         } else
112.             System.out.println("No wire from " + i + " to " + start);
113.     }
114. }
115.
116. public static void displayGraph()
117. {
118.     for (int i = 0; i < n; i++)
119.     {

```

```
120.     System.out.print(i + ": ");
121.     ListIterator nbrs = graph[i].listIterator(0);
122.     while (nbrs.hasNext())
123.     {
124.         Node nd = (Node) nbrs.next();
125.         System.out.print(nd.label + "(" + nd.weight + ") ");
126.     }
127.     System.out.println();
128. }
129. }
130. }
```

Output:

advertisement

```
$ javac ShortestPath.java
$ java ShortestPath
Enter the number of components and wires in a circuit:
4 3
Mention the wire between components and its length:
0 1 2
1 3 3
1 2 2
Enter the start and end for which length is to be minimized:
0 1
0: 1(2)
1: 0(2) 3(3) 2(2)
2: 1(2)
3: 1(3)
Wire from 0 to 0 with length 0:
Wire from 1 to 0 with length 2: 0
Wire from 2 to 0 with length 4: 1 0
Wire from 3 to 0 with length 5: 1 0
```

194. Java Program to Implement the Schonhage-Strassen Algorithm for Multiplication of Two Numbers

[« Prev](#)

[Next »](#)

This is a sample program to multiply two given numbers using Schonhage-Strassen Algorithm. Suppose we are multiplying two numbers like 123 and 456 using long multiplication with base B digits, but without performing any carrying. The result might look something like this:

0	1	2	3
x	4	5	6
<hr/>			
00	00	06	12
00	05	10	15
04	08	12	00
<hr/>			
04	13	28	27
			18

This sequence (4, 13, 28, 27, 18) is called the acyclic or linear convolution of the two original sequences (1,2,3) and (4,5,6). Once you have the acyclic convolution of two sequences, computing the product of the original numbers is easy: you just perform the carrying (for example, in the rightmost column, you'd keep the 8 and add the 1 to the column containing 27). In the example this yields the correct product 56088.

Here is the source code of the Java Program to Implement the Schonhage-Strassen Algorithm for Multiplication of Two Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the multiplication of the two numbers using Schonhage-Strassen Algorithm
2. import java.util.Scanner;
3.
4. public class Schonhage_Strassen_Algorithm
5. {
6.     public static int noOfDigit(long a)
7.     {
8.         int n=0;
9.         while(a>0)
10.        {
11.            a /= 10;
12.            n++;
13.        }
14.        return n;
15.    }
16.    public static void schonhageStrassenMultiplication(long x, long y, int n, int m)
17.    {
18.
19.        int []linearConvolution = new int[n+m-1];
20.        for(int i=0; i<(n+m-1); i++)
21.            linearConvolution[i] = 0;
22.
23.        long p=x;
24.        for(int i=0; i<m; i++)
25.        {
26.            x = p;
27.            for(int j=0; j<n; j++)
28.            {
29.                linearConvolution[i+j] += (y%10) * (x%10);
30.                x /= 10;
31.            }
32.            y /= 10;
33.        }
34.        System.out.print("The Linear Convolution is: ( ");
35.        for(int i=(n+m-2); i>=0; i--)
36.        {
```

```

37.     System.out.print(linearConvolution[i] + " ");
38. }
39. System.out.println(")");
40.
41. long product = 0;
42. int nextCarry=0, base=1;;
43. for(int i=0; i<n+m-1; i++)
44. {
45.     linearConvolution[i] += nextCarry;
46.     product = product + (base * (linearConvolution[i]%10));
47.     nextCarry = linearConvolution[i]/10;
48.     base *= 10;
49. }
50. System.out.println("The Product of the numbers is: " + product);
51.
52. }
53. public static void main(String args[])
54. {
55.     Scanner input = new Scanner(System.in);
56.     System.out.println("Enter the numbers:");
57.     long a = input.nextLong();
58.     long b = input.nextLong();
59.     int n = noOfDigit(a);
60.     int m = noOfDigit(b);
61.     schonhageStrassenMultiplication(a, b, n, m);
62.
63.     input.close();
64. }
65.

```

Output:

advertisement

```
$ javac Schonhage_Strassen_Algorithm.java
$ java Schonhage_Strassen_Algorithm
```

Enter the numbers:

456

123

The Linear Convolution is: (4 13 28 27 18)

The Product of the numbers is: 56088

195. Java Program to Generate All Possible Combinations Out of a, b, c, d, e

[« Prev](#)

[Next »](#)

This is a java program to generate and print all possible combinations out of a, b, c, d, e. The trick here is to start with one letter combinations, then with two letter combinations and so on.

Here is the source code of the Java Program to Generate All Possible Combinations Out of a, b, c, d, e. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to print all possible combinations out of a, b, c, d, e
2.
3. public class All_Possible_Combinatons
4. {
5.     static void printCombinations(char[] sequence, int N)
6.     {
7.         char[] data = new char[N];
8.         for (int r = 0; r < sequence.length; r++)
9.             combinations(sequence, data, 0, N - 1, 0, r);
10.    }
11.
12.    static void combinations(char[] sequence, char[] data, int start, int end,
13.                           int index, int r)
14.    {
15.
16.        if (index == r)
17.        {
18.            for (int j = 0; j < r; j++)
19.                System.out.print(data[j] + " ");
20.            System.out.println();
21.        }
22.
23.        for (int i = start; i <= end && ((end - i + 1) >= (r - index)); i++)
24.        {
25.            data[index] = sequence[i];
26.            combinations(sequence, data, i + 1, end, index + 1, r);
27.        }
28.    }
29.
30.    public static void main(String args[])
31.    {
32.        char[] sequence = { 'a', 'b', 'c', 'd', 'e' };
33.        System.out.print("The combinations are: ");
34.        printCombinations(sequence, sequence.length);
35.    }
36.}
```

Output:

advertisement

```
$ javac All_Possible_Combinatons.java
$ java All_Possible_Combinatons
```

The combinations are:

```
a
b
c
d
e
a b
a c
a d
a e
```

b c
b d
b e
c d
c e
d e
a b c
a b d
a b e
a c d
a c e
a d e
b c d
b c e
b d e
c d e
a b c d
a b c e
a b d e
a c d e
b c d e

196. Java Program to Implement the Binary Counting Method to Generate Subsets of a Set

[« Prev](#)

[Next »](#)

This is a java program to generate and print all possible subsets using the method of Binary Counting method. The generations of subsets are done using binary numbers. Let there be 3 elements in the set, we generate binary equivalent of $2^3 = 8$ numbers(0-7), where each bit in a number represents the presence/absence of element in the subset. The element is present if bit is 1, absent otherwise. 010 – only second element is present in the subset. Here is the source code of the Java Program to Implement the Binary Counting Method to Generate Subsets of a Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate all subsets of given set of numbers using binary counting method
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Binary_Counting_Subsets
6. {
7.     public static int[] binary(int N)
8.     {
9.         int[] binary = new int[(int) Math.pow(2, N)];
10.        for (int i = 0; i < Math.pow(2, N); i++)
11.        {
12.            int b = 1;
13.            binary[i] = 0;
14.            int num = i;
15.            while (num > 0)
16.            {
17.                binary[i] += (num % 2) * b;
18.                num /= 2;
19.                b = b * 10;
20.            }
21.        }
22.        return binary;
23.    }
24.
25.    public static void main(String args[])
26.    {
27.        Random random = new Random();
28.        Scanner sc = new Scanner(System.in);
29.        System.out.println("Enter the number of elements in the set: ");
30.        int N = sc.nextInt();
31.        int[] sequence = new int[N];
32.        for (int i = 0; i < N; i++)
33.            sequence[i] = Math.abs(random.nextInt(100));
34.
35.        System.out.println("The elements in the set : ");
36.        for (int i = 0; i < N; i++)
37.            System.out.print(sequence[i] + " ");
38.
39.        int[] mask = new int[(int) Math.pow(2, N)];
40.        mask = binary(N);
41.
42.        System.out.println("\nThe permutations are: ");
43.        for (int i = 0; i < Math.pow(2, N); i++)
44.        {
45.            System.out.print("{");
46.            for (int j = 0; j < N; j++)
47.            {
48.                if (mask[i] % 10 == 1)
49.                    System.out.print(sequence[j] + " ");
50.                mask[i] /= 10;
51.            }
52.            System.out.print("}");
53.        }
54.    }
55. }
```

```
51.     }
52.     System.out.println("}");
53.   }
54.   sc.close();
55. }
56.}
```

Output:

advertisement

```
$ javac Binary_Counting_Subsets.java
$ java Binary_Counting_Subsets
```

Enter the number of elements in the set:

5

The elements in the set :

78 35 5 10 15

The permutations are:

```
{ }
{ 78 }
{ 35 }
{ 78 35 }
{ 5 }
{ 78 5 }
{ 35 5 }
{ 78 35 5 }
{ 10 }
{ 78 10 }
{ 35 10 }
{ 78 35 10 }
{ 5 10 }
{ 78 5 10 }
{ 35 5 10 }
{ 78 35 5 10 }
{ 15 }
{ 78 15 }
{ 35 15 }
{ 78 35 15 }
{ 5 15 }
{ 78 5 15 }
{ 35 5 15 }
{ 78 35 5 15 }
{ 10 15 }
{ 78 10 15 }
{ 35 10 15 }
{ 78 35 10 15 }
{ 5 10 15 }
{ 78 5 10 15 }
{ 35 5 10 15 }
{ 78 35 5 10 15 }
```

197. Java Program to Implement Bubble Sort

« [Prev](#)

[Next](#) »

This is a java program to sort the numbers using the Bubble Sort Technique. The algorithm goes with the name, generally used to sort numbers in the ascending order. The smallest numbers bubbles up at each iteration of the sort. The time complexity of the algorithm is $O(n^2)$.

Here is the source code of the Java Program to Implement Bubble Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers using bubble sort
2. import java.util.Random;
3.
4. public class Bubble_Sort
5. {
6.     static int[] sort(int[] sequence)
7.     {
8.         // Bubble Sort
9.         for (int i = 0; i < sequence.length; i++)
10.             for (int j = 0; j < sequence.length - 1; j++)
11.                 if (sequence[j] > sequence[j + 1])
12.                 {
13.                     sequence[j] = sequence[j] + sequence[j + 1];
14.                     sequence[j + 1] = sequence[j] - sequence[j + 1];
15.                     sequence[j] = sequence[j] - sequence[j + 1];
16.                 }
17.         return sequence;
18.     }
19.
20.     static void printSequence(int[] sorted_sequence)
21.     {
22.         for (int i = 0; i < sorted_sequence.length; i++)
23.             System.out.print(sorted_sequence[i] + " ");
24.     }
25.
26.     public static void main(String args[])
27.     {
28.         System.out
29.             .println("Sorting of randomly generated numbers using BUBBLE SORT");
30.         Random random = new Random();
31.         int N = 20;
32.         int[] sequence = new int[N];
33.
34.         for (int i = 0; i < N; i++)
35.             sequence[i] = Math.abs(random.nextInt(1000));
36.
37.         System.out.println("\nOriginal Sequence: ");
38.         printSequence(sequence);
39.
40.         System.out.println("\nSorted Sequence: ");
41.         printSequence(sort(sequence));
42.     }
43. }
```

Output:

advertisement

```
$ javac Binary_Counting_Subsets.java
$ java Binary_Counting_Subsets
```

```
$ javac Bubble_Sort.java
$ java Bubble_Sort
```

Sorting of randomly generated numbers using BUBBLE SORT

Original Sequence:

307 677 574 88 325 851 676 357 172 932 166 450 60 538 964 987 706 690 919 518

Sorted Sequence:

60 88 166 172 307 325 357 450 518 538 574 676 677 690 706 851 919 932 964 987

198. Java Program to Implement Bucket Sort

« [Prev](#)

[Next](#) »

This is a java program to sort the numbers using the Bucket Sort Technique. The algorithm allocates the number of memory locations equal to maximum number and initializes all to zero, then each location is incremented as the numbers appears. The time complexity of the algorithm is O(n).

Here is the source code of the Java Program to Implement Bucket Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers using bucket sort
2. import java.util.Random;
3.
4. public class Bucket_Sort
5. {
6.     static int[] sort(int[] sequence, int maxValue)
7.     {
8.         // Bucket Sort
9.         int[] Bucket = new int[maxValue + 1];
10.        int[] sorted_sequence = new int[sequence.length];
11.
12.        for (int i = 0; i < sequence.length; i++)
13.            Bucket[sequence[i]]++;
14.
15.        int outPos = 0;
16.        for (int i = 0; i < Bucket.length; i++)
17.            for (int j = 0; j < Bucket[i]; j++)
18.                sorted_sequence[outPos++] = i;
19.
20.        return sorted_sequence;
21.    }
22.
23.    static void printSequence(int[] sorted_sequence)
24.    {
25.        for (int i = 0; i < sorted_sequence.length; i++)
26.            System.out.print(sorted_sequence[i] + " ");
27.    }
28.
29.    static int maxValue(int[] sequence)
30.    {
31.        int maxValue = 0;
32.        for (int i = 0; i < sequence.length; i++)
33.            if (sequence[i] > maxValue)
34.                maxValue = sequence[i];
35.        return maxValue;
36.    }
37.
38.    public static void main(String args[])
39.    {
40.        System.out
41.            .println("Sorting of randomly generated numbers using BUCKET SORT");
42.        Random random = new Random();
43.        int N = 20;
44.        int[] sequence = new int[N];
45.
46.        for (int i = 0; i < N; i++)
47.            sequence[i] = Math.abs(random.nextInt(100));
48.
49.        int maxValue = maxValue(sequence);
50.
51.        System.out.println("\nOriginal Sequence: ");
52.        printSequence(sequence);
53.
54.        System.out.println("\nSorted Sequence: ");
55.        printSequence(sort(sequence, maxValue));
```

```
56. }  
57. }
```

Output:

advertisement

```
$ javac Bucket_Sort.java  
$ java Bucket_Sort
```

Sorting of randomly generated numbers using BUCKET SORT

Original Sequence:

```
95 9 95 87 8 81 18 54 57 53 92 15 38 24 8 56 29 69 64 66
```

Sorted Sequence:

```
8 8 9 15 18 24 29 38 53 54 56 57 64 66 69 81 87 92 95 95
```

199. Java Program to Perform the Sorting Using Counting Sort

[« Prev](#)

[Next »](#)

This is a java program to sort the numbers using the Counting Sort Technique. In computer science, counting sort is an algorithm for sorting a collection of objects according to keys that are small integers; that is, it is an integer sorting algorithm. It operates by counting the number of objects that have each distinct key value, and using arithmetic on those counts to determine the positions of each key value in the output sequence. Its running time is linear in the number of items and the difference between the maximum and minimum key values.

Here is the source code of the Java Program to Perform the Sorting Using Counting Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers using counting sort
2. import java.util.Random;
3.
4. public class Counting_Sort
5. {
6.     public static int N = 20;
7.     public static int[] sequence = new int[N];
8.     private static final int MAX_RANGE = 1000000;
9.
10.    public static void sort(int[] arr)
11.    {
12.        int N = arr.length;
13.        if (N == 0)
14.            return;
15.        int max = arr[0], min = arr[0];
16.        for (int i = 1; i < N; i++)
17.        {
18.            if (arr[i] > max)
19.                max = arr[i];
20.            if (arr[i] < min)
21.                min = arr[i];
22.        }
23.        int range = max - min + 1;
24.
25.        if (range > MAX_RANGE)
26.        {
27.            System.out.println("\nError : Range too large for sort");
28.            return;
29.        }
30.
31.        int[] count = new int[range];
32.        for (int i = 0; i < N; i++)
33.            count[arr[i] - min]++;
34.        for (int i = 1; i < range; i++)
35.            count[i] += count[i - 1];
36.        int j = 0;
37.        for (int i = 0; i < range; i++)
38.            while (j < count[i])
39.                arr[j++] = i + min;
40.    }
41.
42.    public static void main(String[] args)
43.    {
44.        System.out.println("Counting Sort Test\n");
45.        Random random = new Random();
46.
47.        for (int i = 0; i < N; i++)
48.            sequence[i] = Math.abs(random.nextInt(100));
49.
50.        System.out.println("Elements before sorting");
```

```
51.     for (int i = 0; i < N; i++)  
52.         System.out.print(sequence[i] + " ");  
53.     System.out.println();  
54.  
55.     sort(sequence);  
56.  
57.     System.out.println("\nElements after sorting ");  
58.     for (int i = 0; i < N; i++)  
59.         System.out.print(sequence[i] + " ");  
60.     System.out.println();  
61. }  
62. }
```

Output:

advertisement

```
$ javac Counting_Sort  
$ java Counting_Sort
```

Counting Sort

Elements before sorting
7 7 41 56 91 65 86 84 70 44 90 38 78 58 34 87 56 16 23 86

Elements after sorting
7 7 16 23 34 38 41 44 56 56 58 65 70 78 84 86 86 87 90 91

200. Java Program to Search Number Using Divide and Conquer with the Aid of Fibonacci Numbers

[« Prev](#)

[Next »](#)

This is a java program to search a number using Fibonacci Sequence. The Fibonacci search technique is a method of searching a sorted array using a divide and conquer algorithm that narrows down possible locations with the aid of Fibonacci numbers. Compared to binary search, Fibonacci search examines locations whose addresses have lower dispersion. Therefore, when the elements being searched have non-uniform access memory storage (i.e., the time needed to access a storage location varies depending on the location previously accessed), the Fibonacci search has an advantage over binary search in slightly reducing the average time needed to access a storage location. The typical example of non-uniform access storage is that of a magnetic tape, where the time to access a particular element is proportional to its distance from the element currently under the tape's head. Note, however, that large arrays not fitting in cache or even in RAM can also be considered as non-uniform access examples. Fibonacci search has a complexity of $O(\log(x))$.

Here is the source code of the Java Program to Search Number Using Divide and Conquer with the Aid of Fibonacci Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element using Fibonacci search
2. import java.util.Scanner;
3.
4. public class Fibonacci_Search
5. {
6.     static int kk = -1, nn = -1;
7.     static int fib[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,
8.         377, 610, 98, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368,
9.         75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309,
10.        3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986,
11.        102334155, 165580141 };
12.
13.     static int fibsearch(int a[], int n, long x)
14.     {
15.         int inf = 0, pos, k;
16.         if (nn != n)
17.         {
18.             k = 0;
19.             while (fib[k] < n)
20.                 k++;
21.             kk = k;
22.             nn = n;
23.         }
24.         else
25.             k = kk;
26.
27.         while (k > 0)
28.         {
29.             pos = inf + fib[--k];
30.             if ((pos >= n) || (x < a[pos]))
31.                 ;
32.             else if (x > a[pos])
33.             {
34.                 inf = pos + 1;
35.                 k--;
36.             }
37.             else
38.                 return pos;
39.         }
40.         return -1;
41.     }
42.
43.     public static void main(String args[])
44.     {
```

```
45. int arr[] = { 2, 3, 45, 56, 67, 78, 89, 99, 100, 101 };
46. int num, pos;
47. Scanner scan = new Scanner(System.in);
48. System.out.println("Enter an element to search: ");
49. num = scan.nextInt();
50. pos = fibsearch(arr, 10, num);
51. if (pos >= 0)
52.     System.out.println("\nElement is at index : "
53.                         + fibsearch(arr, 10, num));
54. else
55.     System.out.println("\nElement NOT found!! ");
56. scan.close();
57. }
58. }
```

Output:

```
advertisement

$ javac Fibonacci_Search.java
$ java Fibonacci_Search

Enter an element to search:
78

Element is at index : 5
```

201. Java Program to Generate All Subsets of a Given Set in the Gray Code Order

« Prev

Next »

This is a java program to generate and print all the subsets using the Gray Code Order. The reflected binary code, also known as Gray code after Frank Gray, is a binary numeral system where two successive values differ in only one bit (binary digit). The gray code equivalent of a binary number is (number >> 1) ^ number, i.e. right-shift the number by one and EX-ORing with the original number.

Here is the source code of the Java Program to Generate All Subsets of a Given Set in the Gray Code Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate all subsets of given set of numbers using gray code order
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Gray_Code_Permutation
6. {
7.     public static int[] grayCode(int N)
8.     {
9.         int[] grayCode = new int[(int) Math.pow(2, N)];
10.        int[] binary = new int[(int) Math.pow(2, N)];
11.
12.        for (int i = 0; i < Math.pow(2, N); i++)
13.            grayCode[i] = (i >> 1) ^ i;
14.
15.        for (int i = 0; i < Math.pow(2, N); i++)
16.        {
17.            int b = 1;
18.            binary[i] = 0;
19.            while (grayCode[i] > 0)
20.            {
21.                binary[i] += (grayCode[i] % 2) * b;
22.                grayCode[i] /= 2;
23.                b = b * 10;
24.            }
25.        }
26.        return binary;
27.    }
28.
29.    public static void main(String args[])
30.    {
31.        Random random = new Random();
32.        Scanner sc = new Scanner(System.in);
33.        System.out.println("Enter the number of elements in the set: ");
34.        int N = sc.nextInt();
35.        int[] sequence = new int[N];
36.        for (int i = 0; i < N; i++)
37.            sequence[i] = Math.abs(random.nextInt(100));
38.
39.        System.out.println("The elements in the set : ");
40.        for (int i = 0; i < N; i++)
41.            System.out.print(sequence[i] + " ");
42.
43.        int[] mask = new int[(int) Math.pow(2, N)];
44.        mask = grayCode(N);
45.
46.        System.out.println("\nThe permutations are: ");
47.        for (int i = 0; i < Math.pow(2, N); i++)
48.        {
49.            System.out.print(" { ");
50.            for (int j = 0; j < N; j++)
51.            {
52.                if (mask[i] % 10 == 1)
```

```
53.         System.out.print(sequence[j] + " ");
54.         mask[i] /= 10;
55.     }
56.     System.out.println("}");
57. }
58. sc.close();
59. }
60.}
```

Output:

advertisement

```
$ javac Gray_Code_Permutation.java
$ java Gray_Code_Permutation
```

Enter the number of elements in the set:

4

The elements in the set :

36 75 15 59

The permutations are:

```
{ }
{ 36 }
{ 36 75 }
{ 75 }
{ 75 15 }
{ 36 75 15 }
{ 36 15 }
{ 15 }
{ 15 59 }
{ 36 15 59 }
{ 36 75 15 59 }
{ 75 15 59 }
{ 75 59 }
{ 36 75 59 }
{ 36 59 }
{ 59 }
```

Enter the number of elements in the set:

3

The elements in the set :

73 36 36

The permutations are:

```
{ }
{ 73 }
{ 73 36 }
{ 36 }
{ 36 36 }
{ 73 36 36 }
{ 73 36 }
{ 36 }
```

202. Java Program to Perform integer Partition for a Specific Case

[« Prev](#)

[Next »](#)

This is a java program to generate and print all the partitions of a number such that when those partition elements are added results in the number itself, plus the partition should be unique. We start with the number, number minus one is the next partition and so on, till all one's are the last partition where we stop.

Here is the source code of the Java Program to Perform integer Partition for a Specific Case. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform integer partition such that every partition is unique
2. import java.util.Scanner;
3.
4. public class Integer_Partition
5. {
6.     public static void print(int[]p, int n)
7.     {
8.         for(int i=0; i<n; i++)
9.             System.out.print(p[i]+" ");
10.            System.out.println();
11.    }
12.    public static void generateUniquePartition(int n)
13.    {
14.        int []p = new int[n];
15.        int k = 0;
16.        p[k] = n;
17.        while(true)
18.        {
19.            print(p, k+1);
20.            int rem_value = 0;
21.            while(k >= 0 && p[k] == 1)
22.            {
23.                rem_value += p[k];
24.                k--;
25.            }
26.            if(k < 0)
27.                return;
28.
29.            p[k]--;
30.            rem_value++;
31.
32.            while(rem_value > p[k])
33.            {
34.                p[k+1] = p[k];
35.                rem_value -= p[k];
36.                k++;
37.            }
38.            p[k+1] = rem_value;
39.            k++;
40.        }
41.    }
42.    public static void main(String args[])
43.    {
44.        System.out.println("Partitioning of a given Integer such that every partition is unique");
45.        System.out.println("Enter the number:");
46.        Scanner sc = new Scanner(System.in);
47.        int n = sc.nextInt();
48.        generateUniquePartition(n);
49.        sc.close();
50.    }
51.
52.}
```

Output:

advertisement

```
$ javac Integer_Partition.java  
$ java Integer_Partition
```

Partitioning of a given Integer such that every partition is unique

Enter the number:

```
6  
6  
5 1  
4 2  
4 1 1  
3 3  
3 2 1  
3 1 1 1  
2 2 2  
2 2 1 1  
2 1 1 1 1  
1 1 1 1 1 1
```

203. Java Program to Find k Numbers Closest to Median of S, Where S is a Set of n Numbers

[« Prev](#)

[Next »](#)

This is a java program to find K such elements given by users, such that those numbers are closer to the median of the given sequence. We first find the median of the sequence and then compare with each element such that the difference between the median and number is minimum, and print such k elements.

Here is the source code of the Java Program to Find k Numbers Closest to Median of S, Where S is a Set of n Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find k numbers closest to median of N numbers
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class K_Close_Numbers_Median
6. {
7.     static int N = 25;
8.     static int[] sequence = new int[N];
9.
10.    public static void sort()
11.    {
12.        int i, j, temp;
13.        for (i = 1; i < N; i++)
14.        {
15.            j = i;
16.            temp = sequence[i];
17.            while (j > 0 && temp < sequence[j - 1])
18.            {
19.                sequence[j] = sequence[j - 1];
20.                j = j - 1;
21.            }
22.            sequence[j] = temp;
23.        }
24.    }
25.
26.    public static int median()
27.    {
28.        if(N%2 == 0)
29.            return ((sequence[N/2-1] + sequence[N/2])/2);
30.        else
31.            return sequence[N/2];
32.    }
33.    public static void main(String args[])
34.    {
35.        Random random = new Random();
36.
37.        for(int i=0; i<N; i++)
38.            sequence[i] = Math.abs(random.nextInt(100));
39.        sort();
40.        System.out.println("The Sequence is: ");
41.        for(int i=0; i<N; i++)
42.            System.out.print(sequence[i] + " ");
43.
44.        int median = median();
45.        System.out.println("\nEnter the number of elements close to median you want: ");
46.        Scanner sc = new Scanner(System.in);
47.        int k = sc.nextInt();
48.        int i, j;
49.        if(N%2 == 0)
50.        {
51.            i = N/2-1;
52.            j = N/2;
```

```

53.    }
54.  else
55.  {
56.    i = N/2-1;
57.    j = N/2+1;
58.  }
59. boolean flag = false; int n;
60. for(n=0; n<k; n++)
61. {
62.   if(median-sequence[i] < sequence[j]-median)
63.   {
64.     System.out.print(sequence[i] + " ");
65.     i--;
66.     if(i == -1)
67.     {
68.       n++;
69.       flag = true;
70.       break;
71.     }
72.   }
73. else
74. {
75.   System.out.print(sequence[j] + " ");
76.   j++;
77.   if(j == N)
78.   {
79.     n++;
80.     break;
81.   }
82. }
83. }
84. while(n < k)
85. {
86.   if(flag == true)
87.   {
88.     System.out.print(sequence[j] + " ");
89.     j++;
90.     n++;
91.   }
92. else
93. {
94.   System.out.print(sequence[i] + " ");
95.   i--;
96.   n++;
97. }
98. }
99. }
100.}

```

Output:

advertisement

```
$ javac K_Close_Number_Median.java
$ java K_Close_Number_Median
```

The Sequence is:

3 6 14 17 21 27 27 35 35 38 38 40 40 41 41 43 55 67 73 77 79 82 82 83 87

Enter the number of elements close to median you want:

5

40 41 41 38 38

204. Java Program to Generate All Possible Subsets with Exactly k Elements in Each Subset

[« Prev](#)

[Next »](#)

This is a java program to generate and print all subsets containing exactly k element, where k is provided by user and is \leq number of elements in the set. We first find the median of the sequence and then compare with each element such that the difference between the median and number is minimum, and print such k elements. We can achieve this by using Binary Counting method, where we generate first $2^k - 1$ numbers. The binary number itself represents a subset with 0 as absent element and 1 as present elements. Here is the source code of the Java Program to Generate All Possible Subsets with Exactly k Elements in Each Subset. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate all subsets containing exactly K elements in it
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class K_Elements_Subsets
6. {
7.     public static void main(String args[])
8.     {
9.         Random random = new Random();
10.        Scanner sc = new Scanner(System.in);
11.        System.out.println("Enter the number of elements in the set: ");
12.        int N = sc.nextInt();
13.        int[] sequence = new int[N];
14.        for (int i = 0; i < N; i++)
15.            sequence[i] = Math.abs(random.nextInt(100));
16.
17.        System.out.println("The elements in the set : ");
18.        for (int i = 0; i < N; i++)
19.            System.out.print(sequence[i] + " ");
20.
21.        System.out.println("\nEnter the number of elements in the subsets: ");
22.        int n = sc.nextInt();
23.
24.        int[] binary = new int[(int) Math.pow(2, N)];
25.        for (int i = 0; i < Math.pow(2, N); i++)
26.        {
27.            int b = 1;
28.            binary[i] = 0;
29.            int num = i, count = 0;
30.            while (num > 0)
31.            {
32.                if (num % 2 == 1)
33.                    count++;
34.                binary[i] += (num % 2) * b;
35.                num /= 2;
36.                b = b * 10;
37.            }
38.            if (count == n)
39.            {
40.                System.out.print("{ ");
41.                for (int j = 0; j < N; j++)
42.                {
43.                    if (binary[i] % 10 == 1)
44.                        System.out.print(sequence[j] + " ");
45.                    binary[i] /= 10;
46.                }
47.                System.out.println("}");
48.            }
49.        }
50.        sc.close();
51.    }
52.}
```

Output:

```
$ javac K_Elements_Subsets.java  
$ java K_Elements_Subsets
```

Enter the number of elements in the set:

6

The elements in the set :

51 36 33 97 48 22

Enter the number of elements in the subsets:

3

```
{ 51 36 33 }  
{ 51 36 97 }  
{ 51 33 97 }  
{ 36 33 97 }  
{ 51 36 48 }  
{ 51 33 48 }  
{ 36 33 48 }  
{ 51 97 48 }  
{ 36 97 48 }  
{ 33 97 48 }  
{ 51 36 22 }  
{ 51 33 22 }  
{ 36 33 22 }  
{ 51 97 22 }  
{ 36 97 22 }  
{ 33 97 22 }  
{ 51 48 22 }  
{ 36 48 22 }  
{ 33 48 22 }  
{ 97 48 22 }
```

Enter the number of elements in the set:

5

The elements in the set :

98 74 66 16 76

Enter the number of elements in the subsets:

2

```
{ 98 74 }  
{ 98 66 }  
{ 74 66 }  
{ 98 16 }  
{ 74 16 }  
{ 66 16 }  
{ 98 76 }  
{ 74 76 }  
{ 66 76 }  
{ 16 76 }
```

205. Java Program to Find kth Largest Element in a Sequence

[« Prev](#)

[Next »](#)

This is a java program to find kth largest element form the given sequence of numbers. We find the kth largest by sorting the sequence first and then returning the element at position N-k, which qualifies as the kth largest element of the sequence.

Here is the source code of the Java Program to Find kth Largest Element in a Sequence. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find kth largest element in randomly generated sequence
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Kth_Largest
6. {
7.     static int N = 20;
8.     static int []sequence = new int[N];
9.     public static void sort()
10.    {
11.        System.out.println("The Sequence is: ");
12.        for(int i=0; i<N; i++)
13.            System.out.print(sequence[i] + " ");
14.        System.out.println();
15.
16.        int i, j, temp;
17.        for (i = 1; i< N; i++)
18.        {
19.            j = i;
20.            temp = sequence[i];
21.            while (j > 0 && temp < sequence[j-1])
22.            {
23.                sequence[j] = sequence[j-1];
24.                j = j-1;
25.            }
26.            sequence[j] = temp;
27.        }
28.    }
29.
30.    public static void main(String args[])
31.    {
32.        Random random = new Random();
33.
34.        for(int i=0; i<N; i++)
35.            sequence[i] = Math.abs(random.nextInt(100));
36.
37.        Scanner sc = new Scanner(System.in);
38.        System.out.println("Enter the kth largest to find");
39.        int k = sc.nextInt();
40.
41.        sort();
42.        System.out.println(k+"th largest element is " + sequence[N-k-1]);
43.        sc.close();
44.    }
45. }
```

Output:

advertisement

```
$ javac Kth_Largest.java
$ java Kth_Largest
```

Enter the kth largest to find

5

The Sequence is:

77 20 91 48 29 55 2 53 29 7 20 91 78 21 87 81 49 53 77 1

5th largest element is 77

206. Java Program to Find kth Smallest Element by the Method of Partitioning the Array

[« Prev](#)

[Next »](#)

This is a java program to find kth smallest element form the given sequence of numbers. This could be solved by using Quick sort algorithm, where we partition around the pivot element, the entire sequence of numbers is broken down to two, we arrange the number such that numbers smaller than pivot is kept in the first sequence and numbers larger than the pivot is kept in the second sequence. During this comparison we find the kth smallest element. Here is the source code of the Java Program to Find kth Smallest Element by the Method of Partitioning the Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find kth smallest element form the randomly generated sequence using partitioning
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Kth_Smallest_Partitioning
6. {
7.     public static int N = 20;
8.     public static int[] A = new int[N];
9.
10.    public static void swap(int dex1, int dex2)
11.    {
12.        int temp = A[dex1];
13.        A[dex1] = A[dex2];
14.        A[dex2] = temp;
15.    }
16.
17.    public static int partition(int start, int end)
18.    {
19.        int i = start + 1;
20.        int j = i;
21.        int pivot = start;
22.        for (; i < end; i++)
23.        {
24.            if (A[i] < A[pivot])
25.            {
26.                swap(i, j);
27.                j++;
28.            }
29.        }
30.        if (j <= end)
31.            swap(pivot, (j - 1));
32.
33.        return j - 1;
34.    }
35.
36.    public static void quick_sort(int start, int end, int K) {
37.        int part;
38.        if (start < end)
39.        {
40.            part = partition(start, end);
41.            if (part == K - 1)
42.                System.out.println("kth smallest element : " + A[part]);
43.            if (part > K - 1)
44.                quick_sort(start, part, K);
45.            else
46.                quick_sort(part + 1, end, K);
47.        }
48.        return;
49.    }
50.
```

```
51. public static void main(String args[])
52. {
53.     Random random = new Random();
54.     for (int i = 0; i < N; i++)
55.         A[i] = random.nextInt(1000);
56.
57.     System.out.println("The original sequence is: ");
58.     for (int i = 0; i < N; i++)
59.         System.out.print(A[i] + " ");
60.     Scanner sc = new Scanner(System.in);
61.     System.out.println("\nEnter the Kth smallest you want to find: ");
62.     int k = sc.nextInt();
63.
64.     quick_sort(0, N, k);
65.     sc.close();
66. }
67. }
```

Output:

advertisement

```
$ javac Kth_Smallest_Partitioning.java
$ java Kth_Smallest_Partitioning
```

```
The original sequence is:
811 30 934 118 942 89 855 917 474 194 630 887 916 997 851 550 917 841 343 202
Enter the Kth smallest you want to find:
3
kth smallest element : 118
```

207. Java Program to Generate All Subsets of a Given Set in the Lexico Graphic Order

[« Prev](#)

[Next »](#)

This is a java program to generate and print all the subsets of a given set as per lexicographical order, here we follow the numerical sequence. First generate all the subsets having only one element, then generate all the subsets having two elements and so on.

Here is the source code of the Java Program to Generate All Subsets of a Given Set in the Lexico-Graphic Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate all permutation of n elements in lexicographic order
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Lexicographic_Permutation
6. {
7.     public static int[] lexicographicOrder(int N)
8.     {
9.         int[] binary = new int[(int) Math.pow(2, N)];
10.        for (int i = 0; i < Math.pow(2, N); i++)
11.        {
12.            int b = 1;
13.            binary[i] = 0;
14.            int num = i;
15.            while (num > 0)
16.            {
17.                binary[i] += (num % 2) * b;
18.                num /= 2;
19.                b = b * 10;
20.            }
21.        }
22.        return binary;
23.    }
24.
25.    public static void main(String args[])
26.    {
27.        Random random = new Random();
28.        Scanner sc = new Scanner(System.in);
29.        System.out.println("Enter the number of elements in the set: ");
30.        int N = sc.nextInt();
31.        int[] sequence = new int[N];
32.        for (int i = 0; i < N; i++)
33.            sequence[i] = Math.abs(random.nextInt(100));
34.
35.        System.out.println("The elements in the set : ");
36.        for (int i = 0; i < N; i++)
37.            System.out.print(sequence[i] + " ");
38.
39.        for (int i = 1; i < N; i++)
40.        {
41.            int j = i;
42.            int temp = sequence[i];
43.            while (j > 0 && temp < sequence[j - 1])
44.            {
45.                sequence[j] = sequence[j - 1];
46.                j = j - 1;
47.            }
48.            sequence[j] = temp;
49.        }
50.
51.        int[] mask = new int[(int) Math.pow(2, N)];
52.        mask = lexicographicOrder(N);
```

```
53.  
54. System.out.println("\nThe permutations are: ");  
55. for (int i = 0; i < Math.pow(2, N); i++)  
56. {  
57.     System.out.print("{ ");  
58.     for (int j = 0; j < N; j++)  
59.     {  
60.         if (mask[i] % 10 == 1)  
61.             System.out.print(sequence[j] + " ");  
62.         mask[i] /= 10;  
63.     }  
64.     System.out.println("}");  
65. }  
66. sc.close();  
67. }  
68. }
```

Output:

advertisement

```
$ javac Lexicographic_Permutation.java  
$ java Lexicographic_Permutation
```

Enter the number of elements in the set:

5

The elements in the set :

19 3 37 7 22

The permutations are:

```
{ }  
{ 3 }  
{ 7 }  
{ 19 }  
{ 22 }  
{ 37 }  
{ 3 7 }  
{ 3 19 }  
{ 7 19 }  
{ 3 22 }  
{ 7 22 }  
{ 19 22 }  
{ 3 37 }  
{ 7 37 }  
{ 19 37 }  
{ 22 37 }  
{ 3 7 19 }  
{ 3 7 22 }  
{ 3 19 22 }  
{ 7 19 22 }  
{ 3 7 37 }  
{ 3 19 37 }  
{ 7 19 37 }  
{ 3 22 37 }  
{ 7 22 37 }  
{ 19 22 37 }  
{ 3 7 22 37 }  
{ 3 19 22 37 }  
{ 7 19 22 37 }  
{ 3 7 19 37 }  
{ 3 7 19 22 }  
{ 3 7 19 22 37 }
```

208. Java Program to Find Maximum Element in an Array using Binary Search

[« Prev](#)

[Next »](#)

This is a java program to find the maximum element using binary search technique. Binary search requires sequence to be sorted. We return the last element of the sequence, which is maximum.

Here is the source code of the Java Program to Find Maximum Element in an Array using Binary Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find maximum element using Binary Search
2. import java.util.Random;
3.
4. public class Maximum_Using_Binary
5. {
6.     static int N = 20;
7.     static int []sequence = new int[N];
8.
9.     public static void sort()
10.    {
11.        int i, j, temp;
12.        for (i = 1; i < N; i++)
13.        {
14.            j = i;
15.            temp = sequence[i];
16.            while (j > 0 && temp < sequence[j-1])
17.            {
18.                sequence[j] = sequence[j-1];
19.                j = j-1;
20.            }
21.            sequence[j] = temp;
22.        }
23.    }
24.
25.    public static void main(String args[])
26.    {
27.        Random random = new Random();
28.
29.        for(int i=0; i<N; i++)
30.            sequence[i] = Math.abs(random.nextInt(100));
31.        System.out.println("The sequence is :");
32.        for(int i=0; i<N; i++)
33.            System.out.print(sequence[i] + " ");
34.
35.        sort();
36.
37.        System.out.println("\nThe maximum element in the sequence is : " + sequence[N-1]);
38.    }
39. }
```

Output:

advertisement

```
$ javac Maximum_Using_Binary.java
$ java Maximum_Using_Binary
```

The sequence is :
40 60 99 69 71 90 33 83 7 79 49 67 24 23 36 46 55 13 98 8
The maximum element in the sequence is : 99

209. Java Program to Find Median of Elements where Elements are Stored in 2 Different Arrays

[« Prev](#)

[Next »](#)

This is a java program to find the median from two different array. To do so we merge the two lists and then sort them, after that we find the median of the sequence. If the total number of elements (N) is odd median is the N/2th element, if its even (N/2 + N/2)/2th element.

Here is the source code of the Java Program to Find Median of Elements where Elements are Stored in 2 Different Arrays. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the median of 2 array
2. import java.util.Random;
3.
4. public class Median_Two_Arrays
5. {
6.     static int N = 10, M = 5;
7.     static int[] sequence1 = new int[N];
8.     static int[] sequence2 = new int[M];
9.     static int[] sequence = new int[N+M];
10.
11.    public static void sort()
12.    {
13.        int i, j, temp;
14.        for (i = 1; i < N+M; i++)
15.        {
16.            j = i;
17.            temp = sequence[i];
18.            while (j > 0 && temp < sequence[j - 1])
19.            {
20.                sequence[j] = sequence[j - 1];
21.                j = j - 1;
22.            }
23.            sequence[j] = temp;
24.        }
25.    }
26.
27.    public static void main(String args[])
28.    {
29.        Random random = new Random();
30.
31.        for(int i=0; i<N; i++)
32.            sequence1[i] = Math.abs(random.nextInt(100));
33.        for(int i=0; i<M; i++)
34.            sequence2[i] = Math.abs(random.nextInt(100));
35.        for(int i=0; i<N; i++)
36.            System.out.print(sequence1[i] + " ");
37.        System.out.println();
38.
39.        for(int i=0; i<M; i++)
40.            System.out.print(sequence2[i] + " ");
41.        System.out.println();
42.
43.
44.        int j=0;
45.        for(int i=0; i<N+M; i++)
46.        {
47.            if(i >= N && j < M)
48.                sequence[i] = sequence2[j++];
49.            else
50.                sequence[i] = sequence1[i];
51.        }
52.    }
}
```

```
53.     sort();
54.
55.     if(N+M % 2 == 0)
56.         System.out.println("The Median is : " + (sequence[(N+M)/2-1]+sequence[(N+M)/2])/2);
57.     else
58.         System.out.println("The Median is : " + sequence[(N+M)/2]);
59. }
60.}
```

Output:

advertisement

```
$ javac Median_Two_Arrays.java
$ java Median_Two_Arrays
```

```
92 53 68 15 17 23 95 47 46 61
```

```
63 62 48 66 26
```

```
The Median is : 53
```

210. Java Program to Find Minimum Element in an Array using Linear Search

[« Prev](#)

[Next »](#)

This is a java program to find the minimum element of the sequence using the technique of linear search. First we assign min equal to the first element of the sequence, then we go on exploring each element of the sequence and compare it with the min element, if less we update min.

Here is the source code of the Java Program to Find Minimum Element in an Array using Linear Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the minimum element using the technique of Sequential Search
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Minimum_Using_Sequential
6. {
7.     static int N = 20;
8.     static int []sequence = new int[N];
9.
10.    public static int minSequential()
11.    {
12.        int min = sequence[0];
13.        for(int i=0; i<N; i++)
14.            if(min > sequence[i])
15.                min = sequence[i];
16.
17.        return min;
18.    }
19.    public static void main(String args[])
20.    {
21.        Random random = new Random();
22.
23.        for(int i=0; i<N; i++)
24.            sequence[i] = Math.abs(random.nextInt(100));
25.        System.out.println("The sequence is :");
26.        for(int i=0; i<N; i++)
27.            System.out.print(sequence[i] + " ");
28.
29.        Scanner sc = new Scanner(System.in);
30.        System.out.println("\nThe minimum element in the sequence is : " + minSequential());
31.        sc.close();
32.    }
33. }
```

Output:

advertisement

```
$ javac Minimum_Using_Sequential.java
$ java Minimum_Using_Sequential
```

The sequence is :

33 43 61 93 97 31 53 62 58 87 68 61 16 52 12 29 27 63 68 22

The minimum element in the sequence is : 12

211. Java Program to Find the Number of Ways to Write a Number as the Sum of Numbers Smaller than Itself

[« Prev](#)

[Next »](#)

This is a java program to find the number of ways to write a given number as sum of numbers less than the number itself. We start with the number, number minus one is the next partition and so on, till all one's are the last partition where we stop.

Here is the source code of the Java Program to Find the Number of Ways to Write a Number as the Sum of Numbers Smaller than Itself. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the number of ways to write a number as a sum of smaller than the number itself
2. import java.util.Scanner;
3.
4. public class NumberOf_Unique_Partitions
5. {
6.     public static void print(int[] p, int n, int count)
7.     {
8.         for (int i = 0; i < n; i++)
9.             System.out.print(p[i] + " ");
10.            System.out.println();
11.            int j;
12.            for (j = 0; j < n; j++)
13.            {
14.                if (p[j] == 1)
15.                    continue;
16.                else
17.                    break;
18.            }
19.            if (j == n)
20.                System.out
21.                    .println("The number of ways to write a number as a sum of number smaller than itself is :"
22.                            + (count - 1));
23.    }
24.
25.    public static void generateUniquePartition(int n)
26.    {
27.        int[] p = new int[n];
28.        int k = 0, count = 0;
29.        p[k] = n;
30.        while (true)
31.        {
32.            count++;
33.            print(p, k + 1, count);
34.            int rem_value = 0;
35.            while (k >= 0 && p[k] == 1)
36.            {
37.                rem_value += p[k];
38.                k--;
39.            }
40.            if (k < 0)
41.                return;
42.
43.            p[k]--;
44.            rem_value++;
45.
46.            while (rem_value > p[k])
47.            {
48.                p[k + 1] = p[k];
49.                rem_value -= p[k];
```

```
50.         k++;
51.     }
52.     p[k + 1] = rem_value;
53.     k++;
54. }
55. }
56.
57. public static void main(String args[])
58. {
59.     System.out.println("Unique Partitioning of a given number");
60.     System.out.println("Enter the number:");
61.     Scanner sc = new Scanner(System.in);
62.     int n = sc.nextInt();
63.     generateUniquePartition(n);
64.     sc.close();
65. }
66. }
```

Output:

advertisement

```
$ javac NumberOf_Uncque_Partitions.java
$ java NumberOf_Uncque_Partitions
```

Unique Partitioning of a given number

Enter the number:

```
6
6
5 1
4 2
4 1 1
3 3
3 2 1
3 1 1 1
2 2 2
2 2 1 1
2 1 1 1 1
1 1 1 1 1 1
```

The number of ways to write as a sum of number smaller than itself is :10

212. Java Program to Permute All Letters of an Input String

[« Prev](#)

[Next »](#)

This is the Java Program to Print all Possible Permutations of a String.

Problem Description

Given a string, print all the different words that can be formed by permuting its letters.

Example:

String str = "ABC";

advertisement

Output =

ABC
ACB
BAC
BCA
CAB
CBC

Problem Solution

The solution to this problem is based on the programming paradigm **backtracking**. The idea used here keep a character constant and find the permutations of the remaining (n-1) characters. However, in case of strings having repeated characters, some strings are printed repeatedly.

Program/Source Code

Here is the source code of the Java Program to Print all Possible Permutations of a String. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

advertisement

```
1.  
2. //Java Program to Print all Possible Permutations of a String  
3.  
4. import java.io.BufferedReader;  
5. import java.io.InputStreamReader;  
6.  
7. public class PermutationsOfAString {  
8.     // Function to swap two characters  
9.     static String swap(String str, int i, int j){  
10.         char ch;  
11.         char[] array = str.toCharArray();  
12.         ch = array[i];  
13.         array[i] = array[j];  
14.         array[j] = ch;  
15.         return String.valueOf(array);  
16.     }  
17.     // Function to print all the permutations of the string  
18.     static void permute(String str,int low,int high){  
19.         if(low == high)  
20.             System.out.println(str);  
21.  
22.         int i;  
23.         for(i = low; i<=high; i++){
```

```

24.     str = swap(str,low,i);
25.     permute(str, low+1,high);
26.     str = swap(str,low,i);
27.   }
28. }
29. // Function to read user input
30. public static void main(String[] args) {
31.     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
32.     String str;
33.     System.out.println("Enter a string");
34.     try{
35.         str = br.readLine();
36.     }catch (Exception e){
37.         System.out.println("An error occurred");
38.         return;
39.     }
40.     System.out.println("All the possible permutations of "+ str + "are");
41.     permute(str,0,str.length()-1);
42.   }
43.

```

Program Explanation

1. In function swap(), firstly the string is converted into a character array.
2. Secondly, the characters at i^{th} and j^{th} indexes are interchanged and the character array is finally converted back to string.
3. In function permute(), firstly the low and high indexes are compared if($\text{low} == \text{high}$), to see if they are equal.
4. If it is, that means we are processing the last character and no more permutations can be generated. So the string is printed.
5. The loop for($i = \text{low}; i \leq \text{high}; i++$) is used to generate all the permutations of the string.
6. Firstly, the character at low and i are swapped using $\text{swap}(\text{str}, \text{low}, i)$. Then keeping this character constant, all the other permutations are generated using $\text{permute}(\text{arr}, \text{low}+1, \text{high})$.
7. Finally, the initially swapped character is swapped back to its original position.

Time Complexity: $O(n*(n!))$ where n is the length of the string.

advertisement

Runtime Test Cases

Case 1 (Simple Test Case):

Enter a string
 ABC
 All the possible permutations of ABC are
 ABC
 ACB
 BAC
 BCA
 CBA
 CAB

Case 2 (Simple Test Case - another example):

Enter a string
 ABCA
 All the possible permutations of ABCA are
 ABCA
 ABAC
 ACBA
 ACAB
 AACB
 AABC
 BACA
 BAAC
 BCAA

BCAA
BACA
BAAC
CBAA
CBAA
CABA
CAAB
CAAB
CABA
ABCA
ABAC
ACBA
ACAB
AACB
AABC

213. Java Program to Generate All Possible Combinations of a Given List of Numbers

[« Prev](#)

[Next »](#)

This is a java program to generate and print all the permutation of the Numbers. User first enters the element in the set and then actual elements. The notion of permutation relates to the act of permuting, or rearranging, members of a set into a particular sequence or order (unlike combinations, which are selections that disregard order). For example, there are six permutations of the set {1,2,3}, namely (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), and (3,2,1). Here is the source code of the Java Program to Generate All Possible Combinations of a Given List of Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform all permutation of given list of numbers of a specific length
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Permute_All_List_Numbers
6. {
7.     static void permute(int[] a, int k)
8.     {
9.         if (k == a.length)
10.        {
11.            for (int i = 0; i < a.length; i++)
12.            {
13.                System.out.print(" [" + a[i] + "] ");
14.            }
15.            System.out.println();
16.        }
17.        else
18.        {
19.            for (int i = k; i < a.length; i++)
20.            {
21.                int temp = a[k];
22.                a[k] = a[i];
23.                a[i] = temp;
24.
25.                permute(a, k + 1);
26.
27.                temp = a[k];
28.                a[k] = a[i];
29.                a[i] = temp;
30.            }
31.        }
32.    }
33.
34.    public static void main(String args[])
35.    {
36.        Random random = new Random();
37.        Scanner sc = new Scanner(System.in);
38.        System.out.println("Enter the length of list: ");
39.        int N = sc.nextInt();
40.        int[] sequence = new int[N];
41.
42.        for (int i = 0; i < N; i++)
43.            sequence[i] = Math.abs(random.nextInt(100));
44.
45.        System.out.println("The original sequence is: ");
46.        for (int i = 0; i < N; i++)
47.            System.out.print(sequence[i] + " ");
48.
49.        System.out.println("\nThe permuted sequences are: ");
50.        permute(sequence, 0);
```

```
51.  
52.    sc.close();  
53. }  
54.}
```

Output:

advertisement

```
$ java Permute_All_List_Numbers.java  
$ java Permute_All_List_Numbers
```

Enter the length of list:

3

The original sequence is:

15 61 16

The permuted sequences are:

```
[15] [61] [16]  
[15] [16] [61]  
[61] [15] [16]  
[61] [16] [15]  
[16] [61] [15]  
[16] [15] [61]
```

Enter the length of list:

4

The original sequence is:

50 98 4 61

The permuted sequences are:

```
[50] [98] [4] [61]  
[50] [98] [61] [4]  
[50] [4] [98] [61]  
[50] [4] [61] [98]  
[50] [61] [4] [98]  
[50] [61] [98] [4]  
[98] [50] [4] [61]  
[98] [50] [61] [4]  
[98] [4] [50] [61]  
[98] [4] [61] [50]  
[98] [61] [4] [50]  
[98] [61] [50] [4]  
[4] [98] [50] [61]  
[4] [98] [61] [50]  
[4] [50] [98] [61]  
[4] [50] [61] [98]  
[4] [61] [50] [98]  
[4] [61] [98] [50]  
[61] [98] [4] [50]  
[61] [98] [50] [4]  
[61] [4] [98] [50]  
[61] [4] [50] [98]  
[61] [50] [4] [98]  
[61] [50] [98] [4]
```

214. Java Program to Find the Mode in a Data Set

« [Prev](#)

[Next](#) »

This is a java program to find the mode of a set. The mode of a set is defined as the highest occurring element in the set. We count the occurrence of each of the element and print the element whose count is highest.

Here is the source code of the Java Program to Find the Mode in a Data Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the mode for a given sequence of numbers
2. import java.util.Random;
3.
4. public class Mode
5. {
6.     static int N = 20;
7.     static int[] sequence = new int[N];
8.
9.     public static int mode()
10.    {
11.        int maxValue = 0, maxCount = 0;
12.
13.        for (int i = 0; i < sequence.length; ++i)
14.        {
15.            int count = 0;
16.            for (int j = 0; j < sequence.length; ++j)
17.            {
18.                if (sequence[j] == sequence[i])
19.                    ++count;
20.            }
21.            if (count > maxCount)
22.            {
23.                maxCount = count;
24.                maxValue = sequence[i];
25.            }
26.        }
27.
28.        return maxValue;
29.    }
30.
31.    public static void main(String args[])
32.    {
33.        Random random = new Random();
34.
35.        for (int i = 0; i < N; i++)
36.            sequence[i] = Math.abs(random.nextInt(100));
37.
38.        System.out.println("The set of numbers are:");
39.        for (int i = 0; i < N; i++)
40.            System.out.print(sequence[i] + " ");
41.
42.        System.out.println("\nThe mode of the set is: " + mode());
43.    }
44.}
```

Output:

advertisement

```
$ javac Mode.java
$ java Mode
```

The set of numbers are:
85 3 80 56 37 47 13 11 94 38 6 12 10 31 52 67 81 98 43 37
The mode of the set is: 37

215. Java Program to Implement Sorting of Less than 100 Numbers in O(n) Complexity

[« Prev](#)

[Next »](#)

This is a java program to sort an elements in order n time. Bucket sort can be used to achieve this goal. Bucket sort is O(n) algorithm in time but takes more space, than the normal algorithms.

Here is the source code of the Java Program to Implement Sorting of Less than 100 Numbers in O(n) Complexity. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers less than 100 in O(n) time
2. //Bucket sort is O(n) algorithm
3. import java.util.Random;
4.
5. public class Order_n_Sorting_Algorithm
6. {
7.     static int[] sort(int[] sequence, int maxValue)
8.     {
9.         // Bucket Sort
10.        int[] Bucket = new int[maxValue + 1];
11.        int[] sorted_sequence = new int[sequence.length];
12.
13.        for (int i = 0; i < sequence.length; i++)
14.            Bucket[sequence[i]]++;
15.
16.        int outPos = 0;
17.        for (int i = 0; i < Bucket.length; i++)
18.            for (int j = 0; j < Bucket[i]; j++)
19.                sorted_sequence[outPos++] = i;
20.
21.        return sorted_sequence;
22.    }
23.
24.    static void printSequence(int[] sorted_sequence)
25.    {
26.        for (int i = 0; i < sorted_sequence.length; i++)
27.            System.out.print(sorted_sequence[i] + " ");
28.    }
29.
30.    static int maxValue(int[] sequence)
31.    {
32.        int maxValue = 0;
33.        for (int i = 0; i < sequence.length; i++)
34.            if (sequence[i] > maxValue)
35.                maxValue = sequence[i];
36.        return maxValue;
37.    }
38.
39.    public static void main(String args[])
40.    {
41.        System.out
42.            .println("Sorting of randomly generated numbers using O(n) BUCKET SORT algorithm");
43.        Random random = new Random();
44.        int N = 25;
45.        int[] sequence = new int[N];
46.
47.        for (int i = 0; i < N; i++)
48.            sequence[i] = Math.abs(random.nextInt(100));
49.
50.        int maxValue = maxValue(sequence);
51.
52.        System.out.println("\nOriginal Sequence: ");
53.        printSequence(sequence);
```

```
54.  
55.     System.out.println("\nSorted Sequence: ");  
56.     printSequence(sort(sequence, maxValue));  
57. }  
58.  
59.}
```

Output:

advertisement

```
$ javac Order_n_Sorting_Algorithm.java  
$ java Order_n_Sorting_Algorithm
```

Sorting of randomly generated numbers using O(n) BUCKET SORT algorithm

Original Sequence:

43 50 28 1 80 8 77 92 55 44 15 42 47 98 44 12 78 36 73 57 86 36 11 35 51

Sorted Sequence:

1 8 11 12 15 28 35 36 36 42 43 44 44 47 50 51 55 57 73 77 78 80 86 92 98

216. Java Program to Generate All Pairs of Subsets Whose Union Make the Set

[« Prev](#)

[Next »](#)

This is a java program to generate and print all the pair of subsets whose union makes the original set.

Here is the source code of the Java Program to Generate All Pairs of Subsets Whose Union Make the Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate all pair of subsets whose union results the original set
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Pair_Subset_Union_Set
6. {
7.     public static int[] binary(int N)
8.     {
9.         int[] binary = new int[(int) Math.pow(2, N)];
10.        for (int i = 0; i < Math.pow(2, N); i++)
11.        {
12.            int b = 1;
13.            binary[i] = 0;
14.            int num = i;
15.            while (num > 0)
16.            {
17.                binary[i] += (num % 2) * b;
18.                num /= 2;
19.                b = b * 10;
20.            }
21.        }
22.        return binary;
23.    }
24.
25.    public static void main(String args[])
26.    {
27.        Random random = new Random();
28.        Scanner sc = new Scanner(System.in);
29.        System.out.println("Enter the number of elements in the set: ");
30.        int N = sc.nextInt();
31.        int[] sequence = new int[N];
32.        for (int i = 0; i < N; i++)
33.            sequence[i] = Math.abs(random.nextInt(100));
34.
35.        System.out.println("The elements in the set : ");
36.        for (int i = 0; i < N; i++)
37.            System.out.print(sequence[i] + " ");
38.
39.        int[] mask = new int[(int) Math.pow(2, N)];
40.        mask = binary(N);
41.
42.        System.out
43.            .println("\nThe pair of permutations whose union is original set are: ");
44.        for (int i = 0; i < (Math.pow(2, N) / 2); i++)
45.        {
46.            System.out.print("{ ");
47.            for (int j = 0; j < N; j++)
48.            {
49.                if (mask[i] % 10 == 1)
50.                    System.out.print(sequence[j] + " ");
51.                mask[i] /= 10;
52.            }
53.            System.out.print("} and ");
```

```
54.  
55.     System.out.print("{ ");  
56.     for (int j = 0; j < N; j++)  
57.     {  
58.         if (mask[(int) Math.pow(2, N) - 1 - i] % 10 == 1)  
59.             System.out.print(sequence[j] + " ");  
60.         mask[(int) Math.pow(2, N) - 1 - i] /= 10;  
61.     }  
62.     System.out.println("}");  
63. }  
64. sc.close();  
65. }  
66. }
```

Output:

advertisement

```
$ javac Pair_Subset_Union_Set.java  
$ java Pair_Subset_Union_Set
```

Enter the number of elements in the set:

5

The elements in the set :

3 47 97 79 8

The pair of permutations whose union is original set are:

```
{ } and { 3 47 97 79 8 }  
{ 3 } and { 47 97 79 8 }  
{ 47 } and { 3 97 79 8 }  
{ 3 47 } and { 97 79 8 }  
{ 97 } and { 3 47 79 8 }  
{ 3 97 } and { 47 79 8 }  
{ 47 97 } and { 3 79 8 }  
{ 3 47 97 } and { 79 8 }  
{ 79 } and { 3 47 97 8 }  
{ 3 79 } and { 47 97 8 }  
{ 47 79 } and { 3 97 8 }  
{ 3 47 79 } and { 97 8 }  
{ 97 79 } and { 3 47 8 }  
{ 3 97 79 } and { 47 8 }  
{ 47 97 79 } and { 3 8 }  
{ 3 47 97 79 } and { 8 }
```

Enter the number of elements in the set:

3

The elements in the set :

37 76 87

The pair of permutations whose union is original set are:

```
{ } and { 37 76 87 }  
{ 37 } and { 76 87 }  
{ 76 } and { 37 87 }  
{ 37 76 } and { 87 }
```

217. Java Program to Perform Stooge Sort

[« Prev](#)

[Next »](#)

This is a java program to implement Stooge sort algorithm.

Here is the source code of the Java Program to Perform Stooge Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers using Stooge Sort
2. import java.util.Random;
3.
4. public class Stooge_Sort
5. {
6.
7.     public static int N = 20;
8.     public static int[] sequence = new int[N];
9.
10.    public static int[] stoogeSort(int[] L, int i, int j)
11.    {
12.        if (L[j] < L[i])
13.        {
14.            int swap = L[i];
15.            L[i] = L[j];
16.            L[j] = swap;
17.        }
18.        if ((j - i + 1) >= 3)
19.        {
20.            int t = (j - i + 1) / 3;
21.            stoogeSort(L, i, j - t);
22.            stoogeSort(L, i + t, j);
23.            stoogeSort(L, i, j - t);
24.        }
25.        return L;
26.    }
27.
28.    public static void printSequence(int[] sorted_sequence)
29.    {
30.        for (int i = 0; i < sorted_sequence.length; i++)
31.            System.out.print(sorted_sequence[i] + " ");
32.    }
33.
34.    public static void main(String[] args)
35.    {
36.        Random random = new Random();
37.        System.out
38.            .println("Sorting of randomly generated numbers using STOOGESORT");
39.
40.        for (int i = 0; i < N; i++)
41.            sequence[i] = Math.abs(random.nextInt(1000));
42.
43.        System.out.println("\nOriginal Sequence: ");
44.        printSequence(sequence);
45.
46.        System.out.println("\nSorted Sequence: ");
47.        printSequence(stoogeSort(sequence, 0, sequence.length - 1));
48.    }
49.}
```

Output:

advertisement

\$ javac Stooge_Sort.java

```
$ java Stooge_Sort
```

Sorting of randomly generated numbers using STOOGE SORT

Original Sequence:

213 931 260 34 184 706 346 849 279 918 781 242 995 2 187 378 634 965 138 843

Sorted Sequence:

2 34 138 184 187 213 242 260 279 346 378 634 706 781 843 849 918 931 965 995

218. Java Program to Implement Multi-Threaded Version of Binary Search Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement Threaded Binary Tree. A threaded binary tree makes it possible to traverse the values in the binary tree via a linear traversal that is more rapid than a recursive in-order traversal. It is also possible to discover the parent of a node from a threaded binary tree, without explicit use of parent pointers or a stack, albeit slowly. This can be useful where stack space is limited, or where a stack of parent pointers is unavailable (for finding the parent pointer via DFS).

Here is the source code of the Java Program to Implement Multi-Threaded Version of Binary Search Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element in a multi-threaded version of binary search tree
2. import java.util.Scanner;
3.
4. class TBSTNode
5. {
6.     int ele;
7.     TBSTNode left, right;
8.     boolean leftThread, rightThread;
9.
10.    public TBSTNode(int ele)
11.    {
12.        this(ele, null, null, true, true);
13.    }
14.
15.    public TBSTNode(boolean leftThread, boolean rightThread)
16.    {
17.        this.ele = Integer.MAX_VALUE;
18.        this.left = this;
19.        this.right = this;
20.        this.leftThread = leftThread;
21.        this.rightThread = rightThread;
22.    }
23.
24.    public TBSTNode(int ele, TBSTNode left, TBSTNode right, boolean leftThread, boolean rightThread)
25.    {
26.        this.ele = ele;
27.        this.left = left;
28.        this.right = right;
29.        this.leftThread = leftThread;
30.        this.rightThread = rightThread;
31.    }
32.}
33.
34.class ThreadedBinarySearchTree
35.{
36.    private TBSTNode root;
37.    public ThreadedBinarySearchTree ()
38.    {
39.        root = new TBSTNode(true, false);
40.    }
41.
42.    public void clear()
43.    {
44.        root = new TBSTNode(true, false);
45.    }
46.
47.    public void insert(int ele)
48.    {
```

```

49.     TBSTNode ptr = findNode(root, ele);
50.
51.     if (ptr == null)
52.         return;
53.
54.     if (ptr.ele < ele)
55.     {
56.         TBSTNode nptr = new TBSTNode(ele, ptr, ptr.right, true, true);
57.         ptr.right = nptr;
58.         ptr.rightThread = false;
59.     }
60.     else
61.     {
62.         TBSTNode nptr = new TBSTNode(ele, ptr.left, ptr, true, true);
63.         ptr.left = nptr;
64.         ptr.leftThread = false;
65.     }
66. }
67.
68. public TBSTNode findNode(TBSTNode r, int ele)
69. {
70.     if (r.ele < ele)
71.     {
72.         if (r.rightThread)
73.             return r;
74.         return findNode(r.right, ele);
75.     }
76.     else if (r.ele > ele)
77.     {
78.         if (r.leftThread)
79.             return r;
80.         return findNode(r.left, ele);
81.     }
82.     else
83.         return null;
84. }
85.
86. public boolean search(int ele)
87. {
88.     return findNode(root, ele) == null;
89. }
90.
91. public void inOrder()
92. {
93.     TBSTNode temp = root;
94.     for (;;)
95.     {
96.         temp = insucc(temp);
97.         if (temp == root)
98.             break;
99.         System.out.print(temp.ele + " ");
100.    }
101. }
102.
103. public TBSTNode insucc(TBSTNode tree)
104. {
105.     TBSTNode temp;
106.     temp = tree.right;
107.     if (!tree.rightThread)
108.         while (!temp.leftThread)
109.             temp = temp.left;
110.     return temp;
111. }
112. }
113.
114. public class Threaded_BST

```

```

115. {
116.     public static void main(String[] args)
117.     {
118.         Scanner scan = new Scanner(System.in);
119.
120.         ThreadedBinarySearchTree tbst = new ThreadedBinarySearchTree();
121.         System.out.println("Multi-threaded Binary Search Tree \n");
122.         char ch;
123.
124.         do
125.         {
126.             System.out.println("\nThreaded Binary Search Tree Operations\n");
127.             System.out.println("1. insert ");
128.             System.out.println("2. search");
129.             System.out.println("3. clear");
130.
131.             int choice = scan.nextInt();
132.             switch (choice)
133.             {
134.                 case 1 :
135.                     System.out.println("Enter integer element to insert");
136.                     tbst.insert( scan.nextInt() );
137.                     break;
138.                 case 2 :
139.                     System.out.println("Enter integer element to search");
140.                     System.out.println("Search result : "+ tbst.search( scan.nextInt() ));
141.                     break;
142.                 case 3 :
143.                     System.out.println("\nTree Cleared\n");
144.                     tbst.clear();
145.                     break;
146.                 default :
147.                     System.out.println("Wrong Entry \n ");
148.                     break;
149.             }
150.
151.             System.out.print("\nTree = ");
152.             tbst.inOrder();
153.             System.out.println();
154.
155.             System.out.println("\nDo you want to continue (Type y or n) \n");
156.             ch = scan.next().charAt(0);
157.         } while (ch == 'Y' || ch == 'y');
158.         scan.close();
159.     }
160. }
```

Output:

advertisement

```
$ javac Threaded_BST.java
$ java Threaded_BST
```

Multi-threaded Binary Search Tree

Threaded Binary Search Tree Operations

```

1. insert
2. search
3. clear
1
Enter integer element to insert
23
```

Tree = 23

Do you want to continue (Type y or n)

y

Threaded Binary Search Tree Operations

1. insert

2. search

3. clear

1

Enter integer element to insert

867

Tree = 23 867

Do you want to continue (Type y or n)

y

Threaded Binary Search Tree Operations

1. insert

2. search

3. clear

1

Enter integer element to insert

3

Tree = 3 23 867

Do you want to continue (Type y or n)

y

Threaded Binary Search Tree Operations

1. insert

2. search

3. clear

2

Enter integer element to search

45

Search result : false

Tree = 3 23 867

Do you want to continue (Type y or n)

219. Java Program to Perform Uniform Binary Search

[« Prev](#)

[Next »](#)

This is a java program to perform uniform binary search technique. It uses a lookup table to update a single array index, rather than taking the midpoint of an upper and a lower bound on each iteration; therefore, it is optimized for architectures (such as Knuth's MIX) on which a table look-up is generally faster than an addition and a shift, and many searches will be performed on the same array, or on several arrays of the same length.

Here is the source code of the Java Program to Perform Uniform Binary Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform uniform binary search.
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Uniform_Binary_Search
6. {
7.     static int N = 10;
8.     static int[] sequence = new int[N];
9.     static int[] delta = new int[42];
10.
11.    public static void sort()
12.    {
13.        int i, j, temp;
14.        for (i = 1; i < N; i++)
15.        {
16.            j = i;
17.            temp = sequence[i];
18.            while (j > 0 && temp < sequence[j - 1])
19.            {
20.                sequence[j] = sequence[j - 1];
21.                j = j - 1;
22.            }
23.            sequence[j] = temp;
24.        }
25.    }
26.
27.    public static void make_delta(int N)
28.    {
29.        System.out.println();
30.        int power = 1;
31.        int i = 0;
32.        do
33.        {
34.            int half = power;
35.            power <<= 1;
36.            delta[i] = (N + half) / power;
37.        } while (delta[i++ != 0]);
38.    }
39.
40.    public static int unisearch(int key)
41.    {
42.        int i = delta[0] - 1; /* midpoint of array */
43.        int d = 0;
44.
45.        while (true)
46.        {
47.            if (key == sequence[i])
48.                return i;
49.            else if (delta[d] == 0)
```

```

50.         return -1;
51.     else
52.     {
53.         if (key < sequence[i])
54.             i -= delta[++d];
55.         else
56.             i += delta[++d];
57.     }
58. }
59. }
60.
61. public static void main(String args[])
62. {
63.     Random random = new Random();
64.
65.     for (int i = 0; i < N; i++)
66.         sequence[i] = Math.abs(random.nextInt(100));
67.     System.out.println("The sequence is :");
68.     sort();
69.     for (int i = 0; i < N; i++)
70.         System.out.print(sequence[i] + " ");
71. //sort();
72.     make_delta(N);
73.
74.     System.out.println("Enter the element to be searched: ");
75.     Scanner sc = new Scanner(System.in);
76.     int key = sc.nextInt();
77.     int p = unisearch(key);
78.     if (p > 0)
79.         System.out.println("Element found at position " + p);
80.     else
81.         System.out.println("Element doesn't exist");
82.     sc.close();
83. }
84. }
```

Output:

advertisement

```
$ javac Uniform_Binary_Search.java
$ java Uniform_Binary_Search
```

```
The sequence is :
12 13 20 24 27 32 63 64 74 82
Enter the element to be searched:
24
Element found at position 3
```

```
The sequence is :
0 30 31 32 37 48 51 58 78 87
Enter the element to be searched:
98
Element doesn't exist
```

220. Java Program to Implement Merge Sort Algorithm on Linked List

[« Prev](#)

[Next »](#)

This is a java program to implement merge sort algorithm using linked list.

Here is the source code of the Java Program to Implement Merge Sort Algorithm on Linked List. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java to sort numbers of Linked List using Merge Sort
2. import java.util.Random;
3.
4. class Node
5. {
6.     public int item;
7.     public Node next;
8.
9.     public Node(int val)
10.    {
11.        item = val;
12.    }
13.
14.     public Node()
15.    {}
16.
17.     public void displayNode()
18.    {
19.        System.out.print("[" + item + "] ");
20.    }
21.}
22.
23.class LinkedList
24.{
25.    private Node first;
26.
27.    public LinkedList()
28.    {
29.        first = null;
30.    }
31.
32.    public boolean isEmpty()
33.    {
34.        return (first == null);
35.    }
36.
37.    public void insert(int val)
38.    {
39.        Node newNode = new Node(val);
40.        newNode.next = first;
41.        first = newNode;
42.    }
43.
44.    public void append(Node result)
45.    {
46.        first = result;
47.    }
48.
49.    public void display()
50.    {
51.        Node current = first;
52.        while (current != null)
53.        {
```

```

54.     current.displayNode();
55.     current = current.next;
56.   }
57.   System.out.println("");
58. }
59.
60. public Node extractFirst()
61. {
62.   return first;
63. }
64.
65. public Node MergeSort(Node headOriginal)
66. {
67.   if (headOriginal == null || headOriginal.next == null)
68.     return headOriginal;
69.   Node a = headOriginal;
70.   Node b = headOriginal.next;
71.   while ((b != null) && (b.next != null))
72.   {
73.     headOriginal = headOriginal.next;
74.     b = (b.next).next;
75.   }
76.   b = headOriginal.next;
77.   headOriginal.next = null;
78.   return merge(MergeSort(a), MergeSort(b));
79. }
80.
81. public Node merge(Node a, Node b)
82. {
83.   Node temp = new Node();
84.   Node head = temp;
85.   Node c = head;
86.   while ((a != null) && (b != null))
87.   {
88.     if (a.item <= b.item)
89.     {
90.       c.next = a;
91.       c = a;
92.       a = a.next;
93.     }
94.     else
95.     {
96.       c.next = b;
97.       c = b;
98.       b = b.next;
99.     }
100.   }
101.   c.next = (a == null) ? b : a;
102.   return head.next;
103. }
104. }
105.
106.class Merge_Sort
107.
108. public static void main(String[] args)
109. {
110.   LinkedList object = new LinkedList();
111.   Random random = new Random();
112.   int N = 20;
113.   for (int i = 0; i < N; i++)
114.     object.insert(Math.abs(random.nextInt(100)));
115.
116.   System.out.println("List items before sorting :");
117.   object.display();
118.   object.append(object.MergeSort(object.extractFirst()));
119.   System.out.println("List items after sorting :");

```

```
120.     object.display();
121. }
122. }
```

Output:

advertisement

```
$ javac Merge_Sort.java
$ java Merge_Sort
```

List items before sorting :

```
[41] [11] [6] [13] [41] [62] [26] [46] [71] [16] [52] [57] [23] [81] [25] [4] [20] [75] [68] [51]
```

List items after sorting :

```
[4] [6] [11] [13] [16] [20] [23] [25] [26] [41] [41] [46] [51] [52] [57] [62] [68] [71] [75] [81]
```

221. Java Program to Perform Quick Sort on Large Number of Elements

[« Prev](#)

[Next »](#)

This is a java program to sort the large number of elements using Quick Sort Technique. Quick sort uses a pivot element, where all the elements less than pivot are kept in one list and all the elements greater than pivot are kept in another list, and so on.

Here is the source code of the Java Program to Perform Quick Sort on Large Number of Elements. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort large number of element using Quick Sort
2. import java.util.Random;
3.
4. public class Quick_Sort
5. {
6.     public static int N = 25;
7.     public static int[] sequence = new int[N];
8.
9.     public static void QuickSort(int left, int right)
10.    {
11.        if (right - left <= 0)
12.            return;
13.        else
14.        {
15.            int pivot = sequence[right];
16.            int partition = partitionIt(left, right, pivot);
17.            QuickSort(left, partition - 1);
18.            QuickSort(partition + 1, right);
19.        }
20.    }
21.
22.    public static int partitionIt(int left, int right, long pivot)
23.    {
24.        int leftPtr = left - 1;
25.        int rightPtr = right;
26.        while (true)
27.        {
28.            while (sequence[++leftPtr] < pivot)
29.                ;
30.            while (rightPtr > 0 && sequence[--rightPtr] > pivot)
31.                ;
32.
33.            if (leftPtr >= rightPtr)
34.                break;
35.            else
36.                swap(leftPtr, rightPtr);
37.        }
38.        swap(leftPtr, right);
39.        return leftPtr;
40.    }
41.
42.    public static void swap(int dex1, int dex2)
43.    {
44.        int temp = sequence[dex1];
45.        sequence[dex1] = sequence[dex2];
46.        sequence[dex2] = temp;
47.    }
48.
49.    static void printSequence(int[] sorted_sequence)
50.    {
```

```
51.     for (int i = 0; i < sorted_sequence.length; i++)
52.         System.out.print(sorted_sequence[i] + " ");
53.     }
54.
55.    public static void main(String args[])
56.    {
57.        System.out
58.            .println("Sorting of randomly generated numbers using QUICK SORT");
59.        Random random = new Random();
60.
61.        for (int i = 0; i < N; i++)
62.            sequence[i] = Math.abs(random.nextInt(100));
63.
64.        System.out.println("\nOriginal Sequence: ");
65.        printSequence(sequence);
66.        System.out.println("\nSorted Sequence: ");
67.        QuickSort(0, N - 1);
68.        printSequence(sequence);
69.
70.    }
71.
72.}
```

Output:

advertisement

```
$ javac Quick_Sort.java
$ java Quick_Sort
```

Sorting of randomly generated numbers using QUICK SORT

Original Sequence:

54 22 88 52 43 84 61 75 54 72 7 42 47 15 40 16 46 28 9 48 78 10 89 95 8

Sorted Sequence:

7 8 9 10 15 16 22 28 40 42 43 46 47 48 52 54 54 61 72 75 78 84 88 89 95

222. Java Program to Create a Random Graph Using Random Edge Generation

[« Prev](#)

[Next »](#)

This is a java program to generate a random graph by generating random number of edges. One important thing to note here is, that we need to decide minimum and maximum number of nodes such that all edges get accommodated. Minimum number of vertices is positive solution to $n(n-1) = 2e$, where e is number of edges and maximum number of vertices is $e+1$.

Here is the source code of the Java Program to Create a Random Graph Using Random Edge Generation. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate a random graph using random edge generation
2. import java.util.HashMap;
3. import java.util.LinkedList;
4. import java.util.List;
5. import java.util.Map;
6. import java.util.Random;
7. import java.util.Scanner;
8.
9. public class Random_Edges_Graph
10. {
11.     private Map<Integer, List<Integer>> adjacencyList;
12.
13.     public Random_Edges_Graph(int v)
14.     {
15.         adjacencyList = new HashMap<Integer, List<Integer>>();
16.         for (int i = 1; i <= v; i++)
17.             adjacencyList.put(i, new LinkedList<Integer>());
18.     }
19.
20.     public void setEdge(int to, int from)
21.     {
22.         if (to > adjacencyList.size() || from > adjacencyList.size())
23.             System.out.println("The vertices does not exists");
24.
25.         List<Integer> sls = adjacencyList.get(to);
26.         sls.add(from);
27.         List<Integer> dls = adjacencyList.get(from);
28.         dls.add(to);
29.     }
30.
31.     public List<Integer> getEdge(int to)
32.     {
33.         if (to > adjacencyList.size())
34.         {
35.             System.out.println("The vertices does not exists");
36.             return null;
37.         }
38.         return adjacencyList.get(to);
39.     }
40.
41.     public static void main(String args[])
42.     {
43.         System.out.println("Enter the number of edges: ");
44.
45.         Scanner sc = new Scanner(System.in);
46.         int e = sc.nextInt();
47.         try
48.         {
49.             int minV = (int) Math.ceil((1 + Math.sqrt(1 + 8 * e)) / 2);
```

```

50.     int maxV = e + 1;
51.
52.     Random random = new Random();
53.     int v = Math.abs(random.nextInt(maxV - minV) + minV);
54.     System.out.println("Random graph has "+v+" vertices");
55.
56.     Random_Edges_Graph reg = new Random_Edges_Graph(v);
57.     int count = 1, to, from;
58.     while (count <= e)
59.     {
60.         to = Math.abs(random.nextInt(v + 1 - 1) + 1);
61.         from = Math.abs(random.nextInt(v + 1 - 1) + 1);
62.
63.         reg.setEdge(to, from);
64.         count++;
65.     }
66.
67.     System.out
68.         .println("The Adjacency List Representation of the random graph is: ");
69.
70.     for (int i = 1; i <= v; i++)
71.     {
72.         System.out.print(i + " -> ");
73.         List<Integer> edgeList = reg.getEdge(i);
74.         if (edgeList.size() == 0)
75.             System.out.print("null");
76.         else
77.         {
78.             for (int j = 1;; j++)
79.             {
80.                 if (j != edgeList.size())
81.                     System.out.print(edgeList.get(j - 1) + " -> ");
82.                 else {
83.                     System.out.print(edgeList.get(j - 1));
84.                     break;
85.                 }
86.             }
87.         }
88.         System.out.println();
89.     }
90. }
91. catch (Exception E)
92. {
93.     System.out.println("Something went wrong");
94. }
95. sc.close();
96. }
97.
98.

```

Output:

advertisement

```
$ javac Random_Edges_Graph.java
$ java Random_Edges_Graph
```

Enter the number of edges:

5

Random graph has 4 vertices

The Adjacency List Representation of the random graph is:

```
1 -> 4 -> 4
2 -> 3 -> 3
3 -> 2 -> 4 -> 2
4 -> 1 -> 1 -> 3
```

223. Java Program to Construct a Random Graph by the Method of Random Edge Selection

[« Prev](#)

[Next »](#)

This is a java program to generate a random graph by selecting random number of edges. One important thing to note here is, that we need to decide minimum and maximum number of nodes such that all edges get accommodated. Minimum number of vertices is positive solution to $n(n-1) = 2e$, where e is number of edges and maximum number of vertices is $e+1$.

Here is the source code of the Java Program to Construct a Random Graph by the Method of Random Edge Selection. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to randomly generate a graph using random edge selection
2. import java.util.HashMap;
3. import java.util.LinkedList;
4. import java.util.List;
5. import java.util.Map;
6. import java.util.Random;
7.
8. public class Random_Edges_Graph2
9. {
10.     private Map<Integer, List<Integer>> adjacencyList;
11.
12.     public Random_Edges_Graph2(int v)
13.     {
14.         adjacencyList = new HashMap<Integer, List<Integer>>();
15.         for (int i = 1; i <= v; i++)
16.             adjacencyList.put(i, new LinkedList<Integer>());
17.     }
18.
19.     public void setEdge(int to, int from)
20.     {
21.         if (to > adjacencyList.size() || from > adjacencyList.size())
22.             System.out.println("The vertices does not exists");
23.
24.         List<Integer> sls = adjacencyList.get(to);
25.         sls.add(from);
26.         List<Integer> dls = adjacencyList.get(from);
27.         dls.add(to);
28.     }
29.
30.     public List<Integer> getEdge(int to)
31.     {
32.         if (to > adjacencyList.size())
33.         {
34.             System.out.println("The vertices does not exists");
35.             return null;
36.         }
37.         return adjacencyList.get(to);
38.     }
39.
40.     public static void main(String args[])
41.     {
42.         System.out.println("Random Graph Generation");
43.
44.         Random random = new Random();
45.         int e = Math.abs(random.nextInt(21 - 1) + 1);
46.         try
47.         {
48.             int minV = (int) Math.ceil((1 + Math.sqrt(1 + 8 * e)) / 2);
```

```

49.     int maxV = e + 1;
50.
51.     int v = Math.abs(random.nextInt(maxV - minV) + minV);
52.     System.out.println("Random graph has "+v+" vertices");
53.     System.out.println("Random graph has "+e+" edges");
54.
55.     Random_Edges_Graph2 reg = new Random_Edges_Graph2(v);
56.     int count = 1, to, from;
57.     while (count <= e)
58.     {
59.         to = Math.abs(random.nextInt(v + 1 - 1) + 1);
60.         from = Math.abs(random.nextInt(v + 1 - 1) + 1);
61.
62.         reg.setEdge(to, from);
63.         count++;
64.     }
65.
66.     System.out
67.         .println("The Adjacency List Representation of the random graph is:");
68.
69.     for (int i = 1; i <= v; i++)
70.     {
71.         System.out.print(i + " -> ");
72.         List<Integer> edgeList = reg.getEdge(i);
73.         if (edgeList.size() == 0)
74.             System.out.print("null");
75.         else
76.         {
77.             for (int j = 1;; j++)
78.             {
79.                 if (j != edgeList.size())
80.                     System.out.print(edgeList.get(j - 1) + " -> ");
81.                 else {
82.                     System.out.print(edgeList.get(j - 1));
83.                     break;
84.                 }
85.             }
86.         }
87.         System.out.println();
88.     }
89. }
90. catch (Exception E)
91. {
92.     System.out.println("Something went wrong");
93. }
94. }
95.
96.}

```

Output:

advertisement

```
$ javac Random_Edges_Graph2.java
$ java Random_Edges_Graph2
```

```
Random Graph Generation
Random graph has 4 vertices
Random graph has 5 edges
The Adjacency List Representation of the random graph is:
1 -> 4
2 -> 3 -> 3 -> 4
3 -> 3 -> 3 -> 2 -> 2
4 -> 1 -> 2
```

224. Java Program to Generate Random Partition out of a Given Set of Numbers or Characters

[« Prev](#)

[Next »](#)

This is a java program to generate a random partitioning of a set of characters or numbers in to two sets. Randomly generate an index less than the total number of elements in the set.

Here is the source code of the Java Program to Generate Random Partition out of a Given Set of Numbers or Characters. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform partitioning at random index and generate two sets for given set of numbers or
   characters
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Random_Partition
6. {
7.     public static void main(String args[])
8.     {
9.         Random random = new Random();
10.        Scanner sc = new Scanner(System.in);
11.
12.        int noc = random.nextInt(2);
13.        // if noc is equal to 1 generate numbers
14.        if (noc == 1)
15.        {
16.            int N = 10;
17.            int[] sequence = new int[N];
18.            System.out.print("The Original set of numbers are:\n ");
19.            for (int i = 0; i < N; i++)
20.            {
21.                sequence[i] = Math.abs(random.nextInt(100));
22.                System.out.print(sequence[i] + " ");
23.            }
24.
25.            int partition_index = random.nextInt(11);
26.            System.out.println("\nThe two sequences are: ");
27.            System.out.print("{ ");
28.            for (int i = 0; i < N; i++)
29.            {
30.                if (i == partition_index)
31.                    System.out.print(" } and { ");
32.                System.out.print(sequence[i] + " ");
33.            }
34.            System.out.print("}");
35.            System.out
36.                .println("\nPartitioning around index " + partition_index);
37.
38.        }
39.        // else generate characters
40.        else
41.        {
42.            int N = 10;
43.            char[] sequence = new char[N];
44.            System.out.print("The Original set of characters are:\n ");
45.            for (int i = 0; i < N; i++)
46.            {
47.                sequence[i] = (char) Math.abs(random.nextInt(123 - 97) + 97);
48.                System.out.print(sequence[i] + " ");
49.            }
49.        }
50.    }
51. }
```

```
50.  
51. int partition_index = random.nextInt(11);  
52. System.out.println("\nThe two sequences are: ");  
53. System.out.print("{ ");  
54. for (int i = 0; i < N; i++)  
55. {  
56.     if (i == partition_index)  
57.         System.out.print(" } and { ");  
58.     System.out.print(sequence[i] + " ");  
59. }  
60. System.out.print("}");  
61. System.out  
62.     .println("\nPartitioning around index " + partition_index);  
63.  
64. }  
65. sc.close();  
66. }  
67. }
```

Output:

advertisement

```
$ javac Random_Partition.java  
$ java Random_Partition
```

The Original set of numbers are:
70 13 10 36 78 98 18 64 60 84
The two sequences are:
{ 70 13 10 36 78 98 18 64 } and { 60 84 }
Partitioning around index 8

The Original set of characters are:
n p r e m z y o x p
The two sequences are:
{ n p r e m z } and { y o x p }
Partitioning around index 6

225. Java Program to Generate a Random Subset by Coin Flipping

[« Prev](#)

[Next »](#)

This is a java program to generate a random subset using coin flipping method. Coin is flipped, if is head(1), that element is in the subset else not in the subset.

Here is the source code of the Java Program to Generate a Random Subset by Coin Flipping. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to generate a random subset using coin flipping
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Random_Subset_Coin_Flipping
6. {
7.     static int coinFlip()
8.     {
9.         Random random = new Random();
10.        return random.nextInt(2);
11.    }
12.
13.    public static void main(String args[])
14.    {
15.        Random random = new Random();
16.        Scanner sc = new Scanner(System.in);
17.        System.out.println("Enter the number of elements in the set: ");
18.        int N = sc.nextInt();
19.        int[] sequence = new int[N];
20.
21.        for (int i = 0; i < N; i++)
22.            sequence[i] = Math.abs(random.nextInt(100));
23.
24.        System.out.println("The elements in the set : ");
25.        for (int i = 0; i < N; i++)
26.            System.out.print(sequence[i] + " ");
27.
28.        System.out.print("\n\nThe random subset is: \n{ ");
29.        for (int i = 0; i < N; i++)
30.            if (coinFlip() == 1)
31.                System.out.print(sequence[i] + " ");
32.        System.out.println("}");
33.        sc.close();
34.    }
35.}
```

Output:

advertisement

```
$ javac Random_Subset_Coin_Flipping.java
$ java Random_Subset_Coin_Flipping
```

Enter the number of elements in the set:

10

The elements in the set :

1 44 88 58 8 62 94 59 38 33

The random subset is:

{ 1 44 8 33 }

Enter the number of elements in the set:

3

The elements in the set :

0 42 91

The random subset is:

{ 42 91 }

226. Java Program to Generate a Random UnDirected Graph for a Given Number of Edges

[« Prev](#)

[Next »](#)

This is a java program to generate a random graph using number of edges provided by user. One important thing to note here is, that we need to decide minimum and maximum number of nodes such that all edges get accommodated. Minimum number of vertices is positive solution to $n(n-1) = 2e$, where e is number of edges and maximum number of vertices is $e+1$.

Here is the source code of the Java Program to Generate a Random UnDirected Graph for a Given Number of Edges. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to randomly generate a undirected graph where numbers of edges is given by user
2. import java.util.HashMap;
3. import java.util.LinkedList;
4. import java.util.List;
5. import java.util.Map;
6. import java.util.Random;
7. import java.util.Scanner;
8.
9. public class Random_Undirected_Graph
10. {
11.     private Map<Integer, List<Integer>> adjacencyList;
12.
13.     public Random_Undirected_Graph(int v)
14.     {
15.         adjacencyList = new HashMap<Integer, List<Integer>>();
16.         for (int i = 1; i <= v; i++)
17.             adjacencyList.put(i, new LinkedList<Integer>());
18.     }
19.
20.     public void setEdge(int to, int from)
21.     {
22.         if (to > adjacencyList.size() || from > adjacencyList.size())
23.             System.out.println("The vertices does not exists");
24.
25.         List<Integer> sls = adjacencyList.get(to);
26.         sls.add(from);
27.         List<Integer> dls = adjacencyList.get(from);
28.         dls.add(to);
29.     }
30.
31.     public List<Integer> getEdge(int to)
32.     {
33.         if (to > adjacencyList.size())
34.         {
35.             System.out.println("The vertices does not exists");
36.             return null;
37.         }
38.         return adjacencyList.get(to);
39.     }
40.
41.     public static void main(String args[])
42.     {
43.         System.out.println("Enter the number of edges: ");
44.
45.         Scanner sc = new Scanner(System.in);
46.         int e = sc.nextInt();
47.         try
48.         {
```

```

49.     int minV = (int) Math.ceil((1 + Math.sqrt(1 + 8 * e)) / 2);
50.     int maxV = e + 1;
51.
52.     Random random = new Random();
53.     int v = Math.abs(random.nextInt(maxV - minV) + minV);
54.     System.out.println("Random graph has "+v+" vertices");
55.
56.     Random_Undirected_Graph rug = new Random_Undirected_Graph(v);
57.     int count = 1, to, from;
58.     while (count <= e)
59.     {
60.         to = Math.abs(random.nextInt(v + 1 - 1) + 1);
61.         from = Math.abs(random.nextInt(v + 1 - 1) + 1);
62.
63.         rug.setEdge(to, from);
64.         count++;
65.     }
66.
67.     System.out
68.         .println("The Adjacency List Representation of the graph is:");
69.
70.     for (int i = 1; i <= v; i++)
71.     {
72.         System.out.print(i + " -> ");
73.         List<Integer> edgeList = rug.getEdge(i);
74.         if (edgeList.size() == 0)
75.             System.out.print("null");
76.         else
77.         {
78.             for (int j = 1;; j++)
79.             {
80.                 if (j != edgeList.size())
81.                     System.out.print(edgeList.get(j - 1) + " -> ");
82.                 else {
83.                     System.out.print(edgeList.get(j - 1));
84.                     break;
85.                 }
86.             }
87.         }
88.         System.out.println();
89.     }
90. }
91. catch (Exception E)
92. {
93.     System.out.println("Something went wrong");
94. }
95. sc.close();
96. }
97. }
```

Output:

advertisement

```
$ javac Random_Undirected_Graph.java
$ java Random_Undirected_Graph
```

Enter the number of edges:

15

Random graph has 6 vertices

The Adjacency List Representation of the graph is:

```
1 -> 4 -> 2 -> 3 -> 5
2 -> 2 -> 2 -> 1 -> 5
3 -> 5 -> 6 -> 4 -> 6 -> 1 -> 5 -> 4
4 -> 6 -> 4 -> 4 -> 1 -> 3 -> 3
5 -> 3 -> 6 -> 3 -> 1 -> 2
6 -> 4 -> 3 -> 5 -> 3
```

227. Java Program to Implement Quick Sort Using Randomization

[« Prev](#)

[Next »](#)

This is a java program to perform sorting using Randomized Quick Sort. Randomized Quick Sort randomly selects a pivot element, after selecting pivot standard procedure is to be followed as quick sort.

Here is the source code of the Java Program to Implement Quick Sort Using Randomization. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort numbers using randomized quick sort
2. import java.util.Random;
3.
4. public class Randomized_Quick_Sort
5. {
6.     public static int N = 20;
7.     public static int[] sequence = new int[N];
8.
9.     public static void QuickSort(int left, int right)
10.    {
11.        if (right - left <= 0)
12.            return;
13.        else
14.        {
15.            Random rand = new Random();
16.            int pivotIndex = left + rand.nextInt(right - left + 1);
17.            swap(pivotIndex, right);
18.
19.            int pivot = sequence[right];
20.            int partition = partitionIt(left, right, pivot);
21.            QuickSort(left, partition - 1);
22.            QuickSort(partition + 1, right);
23.        }
24.    }
25.
26.    public static int partitionIt(int left, int right, long pivot)
27.    {
28.        int leftPtr = left - 1;
29.        int rightPtr = right;
30.        while (true)
31.        {
32.            while (sequence[++leftPtr] < pivot)
33.                ;
34.            while (rightPtr > 0 && sequence[--rightPtr] > pivot)
35.                ;
36.
37.            if (leftPtr >= rightPtr)
38.                break;
39.            else
40.                swap(leftPtr, rightPtr);
41.        }
42.        swap(leftPtr, right);
43.        return leftPtr;
44.    }
45.
46.    public static void swap(int dex1, int dex2)
47.    {
48.        int temp = sequence[dex1];
49.        sequence[dex1] = sequence[dex2];
50.        sequence[dex2] = temp;
51.    }
}
```

```
52.  
53. static void printSequence(int[] sorted_sequence)  
54. {  
55.     for (int i = 0; i < sorted_sequence.length; i++)  
56.         System.out.print(sorted_sequence[i] + " ");  
57. }  
58.  
59. public static void main(String args[])  
60. {  
61.     System.out  
62.         .println("Sorting of randomly generated numbers using RANDOMIZED QUICK SORT");  
63.     Random random = new Random();  
64.  
65.     for (int i = 0; i < N; i++)  
66.         sequence[i] = Math.abs(random.nextInt(100));  
67.  
68.     System.out.println("\nOriginal Sequence: ");  
69.     printSequence(sequence);  
70.     System.out.println("\nSorted Sequence: ");  
71.     QuickSort(0, N - 1);  
72.     printSequence(sequence);  
73. }  
74.}
```

Output:

advertisement

```
$ javac Randomized_Quick_Sort.java  
$ java Randomized_Quick_Sort
```

Sorting of randomly generated numbers using RANDOMIZED QUICK SORT

Original Sequence:

98 95 22 64 77 49 11 98 56 63 84 18 9 68 4 69 2 20 68 4

Sorted Sequence:

2 4 4 9 11 18 20 22 49 56 63 64 68 68 69 77 84 95 98 98

228. Java Program to Represent Graph Using Adjacency List

[« Prev](#)

[Next »](#)

This is a java program to represent graph as a adjacency list. Each node will have a linked list consisting of node to which it is connected.

Here is the source code of the Java Program to Represent Graph Using Adjacency List. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a adjacency list
2. import java.util.HashMap;
3. import java.util.LinkedList;
4. import java.util.List;
5. import java.util.Map;
6. import java.util.Scanner;
7.
8. public class Represent_Graph_Adjacency_List
9. {
10.     private Map<Integer, List<Integer>> adjacencyList;
11.
12.     public Represent_Graph_Adjacency_List(int v)
13.     {
14.         adjacencyList = new HashMap<Integer, List<Integer>>();
15.         for (int i = 1; i <= v; i++)
16.             adjacencyList.put(i, new LinkedList<Integer>());
17.     }
18.
19.     public void setEdge(int to, int from)
20.     {
21.         if (to > adjacencyList.size() || from > adjacencyList.size())
22.             System.out.println("The vertices does not exists");
23.
24.         List<Integer> sls = adjacencyList.get(to);
25.         sls.add(from);
26.         List<Integer> dls = adjacencyList.get(from);
27.         dls.add(to);
28.     }
29.
30.     public List<Integer> getEdge(int to)
31.     {
32.         if (to > adjacencyList.size())
33.         {
34.             System.out.println("The vertices does not exists");
35.             return null;
36.         }
37.         return adjacencyList.get(to);
38.     }
39.
40.     public static void main(String args[])
41.     {
42.         int v, e, count = 1, to, from;
43.         Scanner sc = new Scanner(System.in);
44.         Represent_Graph_Adjacency_List glist;
45.         try
46.         {
47.             System.out.println("Enter the number of vertices: ");
48.             v = sc.nextInt();
49.             System.out.println("Enter the number of edges: ");
50.             e = sc.nextInt();
51.         }
```

```

52.     glist = new Represent_Graph_Adjacency_List(v);
53.
54.     System.out.println("Enter the edges in the graph : <to> <from>");
55.     while (count <= e)
56.     {
57.         to = sc.nextInt();
58.         from = sc.nextInt();
59.
60.         glist.setEdge(to, from);
61.         count++;
62.     }
63.
64.     System.out
65.         .println("The Adjacency List Representation of the graph is: ");
66.
67.     for (int i = 1; i <= v; i++)
68.     {
69.         System.out.print(i + "->");
70.         List<Integer> edgeList = glist.getEdge(i);
71.         for (int j = 1;; j++)
72.         {
73.             if (j != edgeList.size())
74.                 System.out.print(edgeList.get(j - 1) + " -> ");
75.             else
76.             {
77.                 System.out.print(edgeList.get(j - 1));
78.                 break;
79.             }
80.         }
81.         System.out.println();
82.     }
83. }
84. catch (Exception E)
85. {
86.     System.out.println("Something went wrong");
87. }
88. sc.close();
89. }
90.

```

Output:

advertisement

```
$ javac Represent_Graph_Adjacency_List.java
$ java Represent_Graph_Adjacency_List
```

Enter the number of vertices:

4 5

Enter the number of edges:

Enter the edges in the graph : <to> <from>

1 2

2 3

3 4

4 1

1 3

The Adjacency List Representation of the graph is:

1 -> 2 -> 4 -> 3

2 -> 1 -> 3

3 -> 2 -> 4 -> 1

4 -> 3 -> 1

229. Java Program to Represent Graph Using Adjacency Matrix

[« Prev](#)

[Next »](#)

This is a java program to represent graph as a adjacency matrix. Nodes are arranged in matrix and at an index of i, j zero is displayed if nodes i and j are not connected, one otherwise.

Here is the source code of the Java Program to Represent Graph Using Adjacency Matrix. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a adjacency matrix
2. import java.util.Scanner;
3.
4. public class Represent_Graph_Adjacency_Matrix
5. {
6.     private final int vertices;
7.     private int[][] adjacency_matrix;
8.
9.     public Represent_Graph_Adjacency_Matrix(int v)
10.    {
11.        vertices = v;
12.        adjacency_matrix = new int[vertices + 1][vertices + 1];
13.    }
14.
15.    public void makeEdge(int to, int from, int edge)
16.    {
17.        try
18.        {
19.            adjacency_matrix[to][from] = edge;
20.        }
21.        catch (ArrayIndexOutOfBoundsException index)
22.        {
23.            System.out.println("The vertices does not exists");
24.        }
25.    }
26.
27.    public int getEdge(int to, int from)
28.    {
29.        try
30.        {
31.            return adjacency_matrix[to][from];
32.        }
33.        catch (ArrayIndexOutOfBoundsException index)
34.        {
35.            System.out.println("The vertices does not exists");
36.        }
37.        return -1;
38.    }
39.
40.    public static void main(String args[])
41.    {
42.        int v, e, count = 1, to = 0, from = 0;
43.        Scanner sc = new Scanner(System.in);
44.        Represent_Graph_Adjacency_Matrix graph;
45.        try
46.        {
47.            System.out.println("Enter the number of vertices: ");
48.            v = sc.nextInt();
49.            System.out.println("Enter the number of edges: ");
50.            e = sc.nextInt();
51.        }
```

```

52.     graph = new Represent_Graph_Adjacency_Matrix(v);
53.
54.     System.out.println("Enter the edges: <to> <from>");
55.     while (count <= e)
56.     {
57.         to = sc.nextInt();
58.         from = sc.nextInt();
59.
60.         graph.makeEdge(to, from, 1);
61.         count++;
62.     }
63.
64.     System.out.println("The adjacency matrix for the given graph is: ");
65.     System.out.print(" ");
66.     for (int i = 1; i <= v; i++)
67.         System.out.print(i + " ");
68.     System.out.println();
69.
70.     for (int i = 1; i <= v; i++)
71.     {
72.         System.out.print(i + " ");
73.         for (int j = 1; j <= v; j++)
74.             System.out.print(graph.getEdge(i, j) + " ");
75.         System.out.println();
76.     }
77.
78. }
79. catch (Exception E)
80. {
81.     System.out.println("Somthing went wrong");
82. }
83.
84. sc.close();
85. }
86.

```

Output:

advertisement

```
$ javac Represent_Graph_Adjacency_Matrix.java
$ java Represent_Graph_Adjacency_Matrix
```

Enter the number of vertices:

5

Enter the number of edges:

7

Enter the edges: <to> <from>

1 1

2 3

3 4

4 5

3 5

1 4

2 4

The adjacency matrix for the given graph is:

1 2 3 4 5

1 1 0 0 1 0

2 0 0 1 1 0

3 0 0 0 1 1

4 0 0 0 0 1

5 0 0 0 0 0

230. Java Program to Represent Graph Using Incidence List

[« Prev](#)

[Next »](#)

This is a java program to represent graph as a incidence list. The incidence matrix of G is a $n \times m$ matrix (b_{ij}), where n and m are the numbers of vertices and edges respectively, such that $b_{ij} = 1$ if the vertex v_i and edge x_j are incident and 0 otherwise. If this relationship is represented by a list, list is known as incidence list.

Here is the source code of the Java Program to Represent Graph Using Incidence List. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a incidence list
2. import java.util.HashMap;
3. import java.util.List;
4. import java.util.Map;
5. import java.util.LinkedList;
6. import java.util.Scanner;
7.
8. public class Represent_Graph_Incidence_List
9. {
10.     private Map<Integer, List<Integer>> incidenceList;
11.     private int v;
12.
13.     public Represent_Graph_Incidence_List(int vertices)
14.     {
15.         v = vertices;
16.         incidenceList = new HashMap<Integer, List<Integer>>();
17.         for (int i = 1; i <= v; i++)
18.             incidenceList.put(i, new LinkedList<Integer>());
19.     }
20.
21.     public void setEdge(int to, int from, int edge_number)
22.     {
23.         List<Integer> slist = incidenceList.get(to);
24.         slist.add(edge_number);
25.         List<Integer> dlist = incidenceList.get(from);
26.         dlist.add(edge_number);
27.
28.     }
29.
30.     public List<Integer> getEdge(int vertex)
31.     {
32.         return incidenceList.get(vertex);
33.     }
34.
35.     public static void main(String args[])
36.     {
37.         int v, e, count = 1, to, from, edgeNumber;
38.         Scanner sc = new Scanner(System.in);
39.         Represent_Graph_Incidence_List glist;
40.         try
41.         {
42.             System.out.println("Enter the number of vertices: ");
43.             v = sc.nextInt();
44.             System.out.println("Enter the number of edges: ");
45.             e = sc.nextInt();
46.
47.             glist = new Represent_Graph_Incidence_List(v);
48.
49.             System.out
50.                 .println("Enter the edges in the graph : <edgenumber> <to> <from>");
```

```

51.     while (count <= e)
52.     {
53.         edgeNumber = sc.nextInt();
54.         to = sc.nextInt();
55.         from = sc.nextInt();
56.
57.         glist.setEdge(to, from, edgeNumber);
58.         count++;
59.     }
60.
61.     System.out
62.         .println("The Incidence List Representation of the graph is:");
63.
64.     System.out.println("Vertex EdgeNumber");
65.     for (int vertex = 1; vertex <= v; vertex++)
66.     {
67.         System.out.print(vertex + "\t");
68.         List<Integer> edgeList = glist.getEdge(vertex);
69.         for (int j = 1;; j++)
70.         {
71.             if (j != edgeList.size())
72.                 System.out.print(edgeList.get(j - 1) + "\t");
73.             else
74.             {
75.                 System.out.print(edgeList.get(j - 1));
76.                 break;
77.             }
78.         }
79.         System.out.println();
80.     }
81. }
82. catch (Exception E)
83. {
84.     System.out.println("Something went wrong");
85. }
86. sc.close();
87. }
88.

```

Output:

advertisement

```
$ javac Represent_Graph_Incidence_List.java
$ java Represent_Graph_Incidence_List
```

Enter the number of vertices:

5

Enter the number of edges:

5

Enter the edges in the graph : <edgenumber> <to> <from>

1 1 2

2 2 4

3 5 4

4 4 3

5 5 1

The Incidence List Representation of the graph is:

Vertex EdgeNumber

1	:	1	5
---	---	---	---

2	:	1	2
---	---	---	---

3	:	4	
---	---	---	--

4	:	2	3	4
---	---	---	---	---

5	:	3	5
---	---	---	---

231. Java Program to Represent Graph Using Incidence Matrix

[« Prev](#)

[Next »](#)

This is a java program to represent graph as a incidence list. The incidence matrix of G is a $n \times m$ matrix (b_{ij}), where n and m are the numbers of vertices and edges respectively, such that $b_{ij} = 1$ if the vertex v_i and edge x_j are incident and 0 otherwise.

Here is the source code of the Java Program to Represent Graph Using Incidence Matrix. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a incidence matrix
2. import java.util.Scanner;
3.
4. public class Represent_Graph_Incidence_Matrix
5. {
6.     private final int rows;
7.     private final int cols;
8.     private int[][] incidence_matrix;
9.
10.    public Represent_Graph_Incidence_Matrix(int v, int e)
11    {
12        rows = v;
13        cols = e;
14        incidence_matrix = new int[rows + 1][cols + 1];
15    }
16.
17.    public void makeEdge(int to, int from, int edge, int edge_number)
18    {
19        try
20        {
21            incidence_matrix[to][edge_number] = edge;
22            incidence_matrix[from][edge_number] = edge;
23        }
24        catch (ArrayIndexOutOfBoundsException index)
25        {
26            System.out.println("The vertices does not exists");
27        }
28    }
29.
30.    public int getEdge(int edge_number, int v)
31    {
32        try
33        {
34            return incidence_matrix[edge_number][v];
35        }
36        catch (ArrayIndexOutOfBoundsException index)
37        {
38            System.out.println("The vertices does not exists");
39        }
40        return -1;
41    }
42.
43.    public static void main(String args[])
44    {
45        int v, e, count = 1, to = 0, from = 0, edge_number;
46        Scanner sc = new Scanner(System.in);
47        Represent_Graph_Incidence_Matrix graph;
48        try
49        {
50            System.out.println("Enter the number of vertices: ");
```

```

51.     v = sc.nextInt();
52.     System.out.println("Enter the number of edges: ");
53.     e = sc.nextInt();
54.
55.     graph = new Represent_Graph_Incidence_Matrix(v, e);
56.
57.     System.out.println("Enter the edges: <edge_number> <to> <from>");
58.     while (count <= e)
59.     {
60.         edge_number = sc.nextInt();
61.         to = sc.nextInt();
62.         from = sc.nextInt();
63.
64.         graph.makeEdge(to, from, 1, edge_number);
65.         count++;
66.     }
67.
68.     System.out.println("The incidence matrix for the given graph is: ");
69.     System.out.print(" ");
70.     for (int i = 1; i <= v; i++)
71.         System.out.print(i + " ");
72.     System.out.println();
73.
74.     for (int i = 1; i <= v; i++)
75.     {
76.         System.out.print(i + " ");
77.         for (int j = 1; j <= v; j++)
78.             System.out.print(graph.getEdge(i, j) + " ");
79.         System.out.println();
80.     }
81.
82. }
83. catch (Exception E)
84. {
85.     System.out.println("Somthing went wrong");
86. }
87.
88. sc.close();
89. }
90.
91. }
```

Output:

advertisement

```
$ javac Represent_Graph_Incidence_Matrix.java
$ java Represent_Graph_Incidence_Matrix
```

Enter the number of vertices:

4

Enter the number of edges:

5

Enter the edges: <edge_number> <to> <from>

1 1 2

2 2 3

3 3 4

4 4 1

1 1 3

The incidence matrix for the given graph is:

1 2 3 4

1 1 0 0 1

2 1 1 0 0

3 1 1 1 0

4 0 0 1 1

232.Java Program to Represent Graph Using Linked List

[« Prev](#)

[Next »](#)

This is a java program to represent graph as a linked list. Each node will have a linked list consisting of node to which it is connected.

Here is the source code of the Java Program to Represent Graph Using Linked List. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a linked list
2. import java.util.HashMap;
3. import java.util.LinkedList;
4. import java.util.List;
5. import java.util.Map;
6. import java.util.Scanner;
7.
8. public class Represent_Graph_Linked_List
9. {
10.     private Map<Integer, List<Integer>> adjacencyList;
11.
12.     public Represent_Graph_Linked_List(int v)
13.     {
14.         adjacencyList = new HashMap<Integer, List<Integer>>();
15.         for (int i = 1; i <= v; i++)
16.             adjacencyList.put(i, new LinkedList<Integer>());
17.     }
18.
19.     public void setEdge(int to, int from)
20.     {
21.         if (to > adjacencyList.size() || from > adjacencyList.size())
22.             System.out.println("The vertices does not exists");
23.
24.         List<Integer> sls = adjacencyList.get(to);
25.         sls.add(from);
26.         List<Integer> dls = adjacencyList.get(from);
27.         dls.add(to);
28.     }
29.
30.     public List<Integer> getEdge(int to)
31.     {
32.         if (to > adjacencyList.size())
33.         {
34.             System.out.println("The vertices does not exists");
35.             return null;
36.         }
37.         return adjacencyList.get(to);
38.     }
39.
40.     public static void main(String args[])
41.     {
42.         int v, e, count = 1, to, from;
43.         Scanner sc = new Scanner(System.in);
44.         Represent_Graph_Linked_List glist;
45.         try
46.         {
47.             System.out.println("Enter the number of vertices: ");
48.             v = sc.nextInt();
49.             System.out.println("Enter the number of edges: ");
50.             e = sc.nextInt();
51.         }
```

```

52.     glist = new Represent_Graph_Linked_List(v);
53.
54.     System.out.println("Enter the edges in the graph : <to> <from>");
55.     while (count <= e)
56.     {
57.         to = sc.nextInt();
58.         from = sc.nextInt();
59.
60.         glist.setEdge(to, from);
61.         count++;
62.     }
63.
64.     System.out
65.         .println("The Linked List Representation of the graph is:");
66.
67.     for (int i = 1; i <= v; i++)
68.     {
69.         System.out.print(i + "->");
70.         List<Integer> edgeList = glist.getEdge(i);
71.         for (int j = 1; j++)
72.         {
73.             if (j != edgeList.size())
74.                 System.out.print(edgeList.get(j - 1) + " -> ");
75.             else
76.             {
77.                 System.out.print(edgeList.get(j - 1));
78.                 break;
79.             }
80.         }
81.         System.out.println();
82.     }
83. }
84. catch (Exception E)
85. {
86.     System.out.println("Something went wrong");
87. }
88. sc.close();
89. }
90. }
```

Output:

advertisement

```
$ javac Represent_Graph_Linked_List.java
$ java Represent_Graph_Linked_List
```

Enter the number of vertices:

5

Enter the number of edges:

4

Enter the edges in the graph : <to> <from>

1 2

1 3

3 5

4 3

The Linked List Representation of the graph is:

1 -> 2 -> 3

2 -> 1

3 -> 1 -> 5 -> 4

4 -> 3

5 -> 3

Java Program to Represent Graph Using 2D Arrays

« [Prev](#)

[Next](#) »

This is a java program to represent graph as a 2D array. Nodes are arranged in matrix and at an index of i, j zero is displayed if nodes i and j are not connected, one otherwise.

Here is the source code of the Java Program to Represent Graph Using 2D Arrays. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to represent graph as a two d array
2. import java.util.Scanner;
3.
4. public class Represent_Graph_TwoD_Array
5. {
6.     private final int vertices;
7.     private int[][] twoD_array;
8.
9.     public Represent_Graph_TwoD_Array(int v)
10.    {
11.        vertices = v;
12.        twoD_array = new int[vertices + 1][vertices + 1];
13.    }
14.
15.    public void makeEdge(int to, int from, int edge)
16.    {
17.        try
18.        {
19.            twoD_array[to][from] = edge;
20.        }
21.        catch (ArrayIndexOutOfBoundsException index)
22.        {
23.            System.out.println("The vertices does not exists");
24.        }
25.    }
26.
27.    public int getEdge(int to, int from)
28.    {
29.        try
30.        {
31.            return twoD_array[to][from];
32.        }
33.        catch (ArrayIndexOutOfBoundsException index)
34.        {
35.            System.out.println("The vertices does not exists");
36.        }
37.        return -1;
38.    }
39.
40.    public static void main(String args[])
41.    {
42.        int v, e, count = 1, to = 0, from = 0;
43.        Scanner sc = new Scanner(System.in);
44.        Represent_Graph_TwoD_Array graph;
45.        try
46.        {
47.            System.out.println("Enter the number of vertices: ");
48.            v = sc.nextInt();
49.            System.out.println("Enter the number of edges: ");
50.            e = sc.nextInt();
51.
52.            graph = new Represent_Graph_TwoD_Array(v);
53.
54.            System.out.println("Enter the edges: <to> <from>");
```

```

55.     while (count <= e)
56.     {
57.         to = sc.nextInt();
58.         from = sc.nextInt();
59.
60.         graph.makeEdge(to, from, 1);
61.         count++;
62.     }
63.
64.     System.out.println("The two d array for the given graph is: ");
65.     System.out.print(" ");
66.     for (int i = 1; i <= v; i++)
67.         System.out.print(i + " ");
68.     System.out.println();
69.
70.     for (int i = 1; i <= v; i++)
71.     {
72.         System.out.print(i + " ");
73.         for (int j = 1; j <= v; j++)
74.             System.out.print(graph.getEdge(i, j) + " ");
75.         System.out.println();
76.     }
77.
78. }
79. catch (Exception E)
80. {
81.     System.out.println("Something went wrong");
82. }
83. sc.close();
84. }
85.

```

Output:

advertisement

```
$ javac Represent_Graph_TwoD_Array.java
$ java Represent_Graph_TwoD_Array
```

Enter the number of vertices:

4

Enter the number of edges:

5

Enter the edges: <to> <from>

1 2

2 3

3 4

1 3

1 4

The two d array for the given graph is:

1 2 3 4

1 0 1 1 1

2 0 0 1 0

3 0 0 0 1

4 0 0 0 0

233.Java Program to Search for an Element in a Binary Search Tree

[« Prev](#)

[Next »](#)

This is a java program to search an element using Binary Search Tree. A regular tree traversal algorithm is implemented to search an element. We start from root, if value to be searched is less than root we traverse left, else we check if its greater we traverse right, else it is equal and return true.

Here is the source code of the Java Program to Search for an Element in a Binary Search Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element in binary search tree
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. /* Class BSTNode */
6. class BSTNode
7. {
8.     BSTNode left, right;
9.     int data;
10.
11.    /* Constructor */
12.    public BSTNode()
13.    {
14.        left = null;
15.        right = null;
16.        data = 0;
17.    }
18.
19.    /* Constructor */
20.    public BSTNode(int n)
21.    {
22.        left = null;
23.        right = null;
24.        data = n;
25.    }
26.
27.    /* Function to set left node */
28.    public void setLeft(BSTNode n)
29.    {
30.        left = n;
31.    }
32.
33.    /* Function to set right node */
34.    public void setRight(BSTNode n)
35.    {
36.        right = n;
37.    }
38.
39.    /* Function to get left node */
40.    public BSTNode getLeft()
41.    {
42.        return left;
43.    }
44.
45.    /* Function to get right node */
46.    public BSTNode getRight()
47.    {
48.        return right;
49.    }
50.
```

```

51. /* Function to set data to node */
52. public void setData(int d)
53. {
54.
55.     data = d;
56. }
57.
58. /* Function to get data from node */
59. public int getData()
60. {
61.     return data;
62. }
63. }
64.
65./* Class BST */
66.class BST
67.{
68.    private BSTNode root;
69.
70. /* Constructor */
71. public BST()
72. {
73.     root = null;
74. }
75.
76. /* Function to check if tree is empty */
77. public boolean isEmpty()
78. {
79.     return root == null;
80. }
81.
82. /* Functions to insert data */
83. public void insert(int data)
84. {
85.     root = insert(root, data);
86. }
87.
88. /* Function to insert data recursively */
89. private BSTNode insert(BSTNode node, int data)
90. {
91.     if (node == null)
92.         node = new BSTNode(data);
93.     else
94.     {
95.         if (data <= node.getData())
96.             node.left = insert(node.left, data);
97.         else
98.             node.right = insert(node.right, data);
99.     }
100.    return node;
101. }
102.
103. /* Functions to delete data */
104. public void delete(int k)
105. {
106.     if (isEmpty())
107.         System.out.println("Tree Empty");
108.     else if (search(k) == false)
109.         System.out.println("Sorry " + k + " is not present");
110.     else
111.     {
112.         root = delete(root, k);
113.         System.out.println(k + " deleted from the tree");
114.     }
115. }
116.
```

```

117. private BSTNode delete(BSTNode root, int k)
118. {
119.     BSTNode p, p2, n;
120.     if (root.getData() == k)
121.     {
122.         BSTNode lt, rt;
123.         lt = root.getLeft();
124.         rt = root.getRight();
125.         if (lt == null && rt == null)
126.             return null;
127.         else if (lt == null)
128.         {
129.             p = rt;
130.             return p;
131.         }
132.         else if (rt == null)
133.         {
134.             p = lt;
135.             return p;
136.         }
137.         else
138.         {
139.             p2 = rt;
140.             p = rt;
141.             while (p.getLeft() != null)
142.                 p = p.getLeft();
143.             p.setLeft(lt);
144.             return p2;
145.         }
146.     }
147.     if (k < root.getData())
148.     {
149.         n = delete(root.getLeft(), k);
150.         root.setLeft(n);
151.     }
152.     else
153.     {
154.         n = delete(root.getRight(), k);
155.         root.setRight(n);
156.     }
157.     return root;
158. }
159.
160. /* Functions to count number of nodes */
161. public int countNodes()
162. {
163.     return countNodes(root);
164. }
165.
166. /* Function to count number of nodes recursively */
167. private int countNodes(BSTNode r)
168. {
169.     if (r == null)
170.         return 0;
171.     else
172.     {
173.         int l = 1;
174.         l += countNodes(r.getLeft());
175.         l += countNodes(r.getRight());
176.         return l;
177.     }
178. }
179.
180. /* Functions to search for an element */
181. public boolean search(int val)
182. {

```

```

183.     return search(root, val);
184. }
185.
186. /* Function to search for an element recursively */
187. private boolean search(BSTNode r, int val)
188. {
189.     boolean found = false;
190.     while ((r != null) && !found)
191.     {
192.         int rval = r.getData();
193.         if (val < rval)
194.             r = r.getLeft();
195.         else if (val > rval)
196.             r = r.getRight();
197.         else
198.         {
199.             found = true;
200.             break;
201.         }
202.         found = search(r, val);
203.     }
204.     return found;
205. }
206.
207. /* Function for inorder traversal */
208. public void inorder()
209. {
210.     inorder(root);
211. }
212.
213. private void inorder(BSTNode r)
214. {
215.     if (r != null)
216.     {
217.         inorder(r.getLeft());
218.         System.out.print(r.getData() + " ");
219.         inorder(r.getRight());
220.     }
221. }
222.
223. /* Function for preorder traversal */
224. public void preorder()
225. {
226.     preorder(root);
227. }
228.
229. private void preorder(BSTNode r)
230. {
231.     if (r != null)
232.     {
233.         System.out.print(r.getData() + " ");
234.         preorder(r.getLeft());
235.         preorder(r.getRight());
236.     }
237. }
238.
239. /* Function for postorder traversal */
240. public void postorder()
241. {
242.     postorder(root);
243. }
244.
245. private void postorder(BSTNode r)
246. {
247.     if (r != null)
248.     {

```

```
249.     postorder(r.getLeft());
250.     postorder(r.getRight());
251.     System.out.print(r.getData() + " ");
252.   }
253. }
254.}
255.
256.public class Search_Element_BST
257.{
258.  public static int N = 20;
259.
260.  public static void main(String args[])
261.  {
262.    Random random = new Random();
263.    BST bst = new BST();
264.    for (int i = 0; i < N; i++)
265.      bst.insert(Math.abs(random.nextInt(100)));
266.
267.    System.out.print("In order traversal of the tree :\n");
268.    bst.inorder();
269.
270.    System.out.println("\nEnter the element to be searched: ");
271.    Scanner sc = new Scanner(System.in);
272.    System.out.println("Search result : " + bst.search(sc.nextInt()));
273.
274.    sc.close();
275.  }
276.}
```

Output:

advertisement

```
$ javac Search_Element_BST.java
$ java Search_Element_BST
```

```
In order traversal of the tree :
3 4 7 17 18 40 46 48 53 54 59 60 74 77 85 91 92 93 95 96
Enter the element to be searched:
51
Search result : false
```

234.Java Program to Perform Searching Using Self-Organizing Lists

[« Prev](#)

[Next »](#)

This is a java program to search an element using Self Organizing lists. A self-organizing list is a list that reorders its elements based on some self-organizing heuristic to improve average access time. The aim of a self-organizing list is to improve efficiency of linear search by moving more frequently accessed items towards the head of the list. A self-organizing list achieves near constant time for element access in the best case. A self-organizing list uses a reorganizing algorithm to adapt to various query distributions at runtime.

Here is the source code of the Java Program to Perform Searching Using Self-Organizing Lists. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element in self organizing lists
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. class SelfOrganizingList
6. {
7.     private int[] list;
8.     private int[] count;
9.     private int size;
10.
11.    public SelfOrganizingList(int listSize)
12.    {
13.        list = new int[listSize];
14.        count = new int[listSize];
15.        size = 0;
16.    }
17.
18.    public boolean isEmpty()
19.    {
20.        return size == 0;
21.    }
22.
23.    public boolean isFull()
24.    {
25.        return size == list.length;
26.    }
27.
28.    public void makeEmpty()
29.    {
30.        int l = list.length;
31.        list = new int[l];
32.        count = new int[l];
33.        size = 0;
34.    }
35.
36.    public int getSize()
37.    {
38.        return size;
39.    }
40.
41.    public void insert(int val)
42.    {
43.        if (isFull())
44.        {
45.            System.out.println("Error : List full!");
46.            return;
47.        }
48.        list[size] = val;
```

```

49.     count[size] = 0;
50.     size++;
51. }
52.
53. public void remove(int pos)
54. {
55.     pos--;
56.     if (pos < 0 || pos >= size)
57.     {
58.         System.out.println("Invalid position ");
59.         return;
60.     }
61.     for (int i = pos; i < size - 1; i++)
62.     {
63.         list[i] = list[i + 1];
64.         count[i] = count[i + 1];
65.     }
66.     size--;
67. }
68.
69. public boolean search(int x)
70. {
71.     boolean searchResult = false;
72.     int pos = -1;
73.     for (int i = 0; i < size; i++)
74.     {
75.         if (list[i] == x) {
76.             searchResult = true;
77.             pos = i;
78.             break;
79.         }
80.     }
81.     if (searchResult)
82.     {
83.         count[pos]++;
84.         int c = count[pos];
85.         for (int i = 0; i < pos; i++)
86.         {
87.             if (count[pos] > count[i])
88.             {
89.                 for (int j = pos; j > i; j--)
90.                 {
91.                     list[j] = list[j - 1];
92.                     count[j] = count[j - 1];
93.                 }
94.                 list[i] = x;
95.                 count[i] = c;
96.                 break;
97.             }
98.         }
99.     }
100.    return searchResult;
101. }
102.
103. public void printList()
104. {
105.     System.out.print("\nList = ");
106.     for (int i = 0; i < size; i++)
107.         System.out.print(list[i] + " ");
108.     System.out.print("\nCount = ");
109.     for (int i = 0; i < size; i++)
110.         System.out.print(count[i] + " ");
111. }
112. }
113.
114. public class Search_Self_Organizing

```

```
115. {  
116. public static void main(String[] args)  
117. {  
118.     Random random = new Random();  
119.     int N = 20;  
120.  
121.     SelfOrganizingList list = new SelfOrganizingList(N);  
122.     for (int i = 0; i < N; i++)  
123.         list.insert(Math.abs(random.nextInt(1000)));  
124.  
125.     Scanner scan = new Scanner(System.in);  
126.     System.out.println("SelfOrganizingList Searching \n");  
127.  
128.     list.printList();  
129.  
130.     System.out.println("\nEnter integer element to search");  
131.     System.out.println("Search Result : " + list.search(scan.nextInt()));  
132.  
133.     scan.close();  
134. }  
135. }
```

Output:

advertisement

```
$ javac Search_Self_Organizing.java  
$ java Search_Self_Organizing
```

SelfOrganizingList Searching

List = 855 318 462 851 373 360 811 401 813 50 291 346 707 118 633 217 715 594 999 99
Count = 0

Enter integer element to search

811

Search Result : true

235.Java Program to Find Second Smallest of n Elements with Given Complexity Constraint

[« Prev](#)

[Next »](#)

This is a java program to find the second smallest element with given complexity. Complexity here is minimum space constraints. Inplace sorting and returning second element help achieving the space constraints.

Here is the source code of the Java Program to Find Second Smallest of n Elements with Given Complexity Constraint. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the second smallest element of N elements with the minimum space complexity constraints
2. import java.util.Random;
3.
4. public class Second_Smallest_Element
5. {
6.     static int kthminimum(int[] sequence, int k)
7.     {
8.         // Bubble Sort for length of sequence minus k times
9.         for (int i = 0; i < (sequence.length - k); i++)
10.             for (int j = 0; j < sequence.length - 1; j++)
11.                 if (sequence[j] > sequence[j + 1])
12.                 {
13.                     sequence[j] = sequence[j] + sequence[j + 1];
14.                     sequence[j + 1] = sequence[j] - sequence[j + 1];
15.                     sequence[j] = sequence[j] - sequence[j + 1];
16.                 }
17.         return sequence[k - 1];
18.     }
19.
20.     public static void main(String args[])
21.     {
22.         Random random = new Random();
23.         int N = 20;
24.         int[] sequence = new int[N];
25.
26.         for (int i = 0; i < N; i++)
27.             sequence[i] = Math.abs(random.nextInt(1000));
28.
29.         System.out.println("Original Sequence: ");
30.         for (int i = 0; i < N; i++)
31.             System.out.print(sequence[i] + " ");
32.
33.         System.out.println("\nSecond smallest element :\n"
34.             + kthminimum(sequence, 2));
35.     }
36. }
```

Output:

advertisement

```
$ javac Second_Smallest_Element.java
$ java Second_Smallest_Element
```

Original Sequence:
459 886 873 766 616 878 122 372 453 876 845 965 477 139 788 861 148 5 894 439
Second smallest element :
122

Original Sequence:

695 213 257 62 315 289 234 90 153 721 192 183 676 373 292 928 57 472 200 177

Second smallest element :

62

236.Java Program to Compare Binary and Sequential Search

[« Prev](#)

[Next »](#)

This is a java program to compare Binary Search and Linear Search algorithms. Following class provides the time required to search an element for both the algorithms

Here is the source code of the Java Program to Compare Binary and Sequential Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This the the java program to compare the sequential and binary search
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Sequential_Binary_Compare
6. {
7.     public static int N = 1000;
8.     public static int[] sequence = new int[N];
9.
10.    public static boolean sequentialSearch(int[] sequence, int key)
11.    {
12.        for (int i = 0; i < sequence.length; i++)
13.            if (sequence[i] == key)
14.                return true;
15.            return false;
16.    }
17.
18.    public static boolean binarySearch(int[] sequence, int key)
19.    {
20.        int low = 0, high = sequence.length - 1;
21.        while (low <= high)
22.        {
23.            int mid = (low + high) / 2;
24.            if (key < sequence[mid])
25.                high = mid - 1;
26.            else if (key > sequence[mid])
27.                low = mid + 1;
28.            else
29.                return true;
30.        }
31.        return false;
32.    }
33.
34.    public static void QuickSort(int left, int right)
35.    {
36.        if (right - left <= 0)
37.            return;
38.        else
39.        {
40.            int pivot = sequence[right];
41.            int partition = partitionIt(left, right, pivot);
42.            QuickSort(left, partition - 1);
43.            QuickSort(partition + 1, right);
44.        }
45.    }
46.
47.    public static int partitionIt(int left, int right, long pivot)
48.    {
49.        int leftPtr = left - 1;
50.        int rightPtr = right;
51.        while (true)
```

```

52. {
53.     while (sequence[++leftPtr] < pivot)
54.         ;
55.     while (rightPtr > 0 && sequence[--rightPtr] > pivot)
56.         ;
57.
58.     if (leftPtr >= rightPtr)
59.         break;
60.     else
61.         swap(leftPtr, rightPtr);
62.     }
63.     swap(leftPtr, right);
64.     return leftPtr;
65. }
66.
67. public static void swap(int dex1, int dex2)
68. {
69.     int temp = sequence[dex1];
70.     sequence[dex1] = sequence[dex2];
71.     sequence[dex2] = temp;
72. }
73.
74. public static void main(String args[])
75. {
76.     Random random = new Random();
77.
78.     for (int i = 0; i < N; i++)
79.         sequence[i] = Math.abs(random.nextInt(100));
80.
81.     Scanner sc = new Scanner(System.in);
82.     System.out.println("Enter the key to be searched: ");
83.     int k = sc.nextInt();
84.
85.     System.out
86.         .println("Time taken to search key using sequential search: ");
87.     long startTime = System.nanoTime();
88.     boolean result = sequentialSearch(sequence, k);
89.     long endTime = System.nanoTime();
90.
91.     if (result == true)
92.         System.out.println("Key found in " + (endTime - startTime)
93.             + " nanoseconds");
94.     else
95.         System.out.println("Key doesn't exist, execution time "
96.             + (endTime - startTime) + " nanoseconds");
97.
98.     System.out.println("Time taken to search key using binary search: ");
99.     QuickSort(0, N - 1);
100.    startTime = System.nanoTime();
101.    result = sequentialSearch(sequence, k);
102.    endTime = System.nanoTime();
103.
104.    if (result == true)
105.        System.out.println("Key found in " + (endTime - startTime)
106.            + " nanoseconds");
107.    else
108.        System.out.println("Key doesn't exist, execution time "
109.            + (endTime - startTime) + " nanoseconds");
110.    sc.close();
111. }
112. }
```

Output:

advertisement

\$ javac Sequential_Binary_Compare.java

```
$ java Sequential_Binary_Compare
```

```
Enter the key to be searched: (N=100)
```

```
85
```

```
Time taken to search key using sequential search:
```

```
Key found in 14696 nanoseconds
```

```
Time taken to search key using binary search:
```

```
Key found in 6680 nanoseconds
```

```
Enter the key to be searched: (N=1000)
```

```
562
```

```
Time taken to search key using sequential search:
```

```
Key doesn't exist, execution time 44422 nanoseconds
```

```
Time taken to search key using binary search:
```

```
Key doesn't exist, execution time 43420 nanoseconds
```

237.Java Program to Perform the Shaker Sort

[« Prev](#)

[Next »](#)

This is a java program to implement Shaker Sort algorithm.

Here is the source code of the Java Program to Perform the Shaker Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to sort the numbers using Shaker Sort
2. import java.util.Random;
3.
4. public class Shaker_Sort
5. {
6.     public static void printSequence(int[] sorted_sequence)
7.     {
8.         for (int i = 0; i < sorted_sequence.length; i++)
9.             System.out.print(sorted_sequence[i] + " ");
10.    }
11.
12.    public static int[] shakerSort(int[] array) {
13.        for (int i = 0; i < array.length/2; i++) {
14.            boolean swapped = false;
15.            for (int j = i; j < array.length - i - 1; j++) {
16.                if (array[j] < array[j+1]) {
17.                    int tmp = array[j];
18.                    array[j] = array[j+1];
19.                    array[j+1] = tmp;
20.                }
21.            }
22.            for (int j = array.length - 2 - i; j > i; j--) {
23.                if (array[j] > array[j-1]) {
24.                    int tmp = array[j];
25.                    array[j] = array[j-1];
26.                    array[j-1] = tmp;
27.                    swapped = true;
28.                }
29.            }
30.            if(!swapped) break;
31.        }
32.        return array;
33.    }
34.    public static void main(String args[])
35.    {
36.        System.out
37.            .println("Sorting of randomly generated numbers using Shaker SORT");
38.        Random random = new Random();
39.        int N = 20;
40.        int[] sequence = new int[N];
41.
42.        for (int i = 0; i < N; i++)
43.            sequence[i] = Math.abs(random.nextInt(100));
44.
45.        System.out.println("\nOriginal Sequence: ");
46.        printSequence(sequence);
47.
48.        System.out.println("\nSorted Sequence: ");
49.        printSequence(shakerSort(sequence));
50.    }
51.
52.}
```

Output:

advertisement

```
$ javac Shaker_Sort.java  
$ java Shaker_Sort
```

Sorting of randomly generated numbers using SHAKER SORT

Original Sequence:

195 853 655 915 364 689 539 684 956 197 67 871 509 662 825 336 540 815 403 876

Sorted Sequence:

956 915 876 871 853 825 815 689 684 662 655 540 539 509 403 364 336 197 195 67

238.Java Program to Implement Shell Sort

« [Prev](#)

[Next](#) »

This is a java program to implement Shell Sort Algorithm.

Here is the source code of the Java Program to Implement Shell Sort. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a program to sort numbers using Shell Sort
2. import java.util.Random;
3.
4. public class Shell_Sort
5. {
6.     public static int N = 20;
7.     public static int[] sequence = new int[N];
8.
9.     public static void shellSort()
10.    {
11.        int increment = sequence.length / 2;
12.        while (increment > 0)
13.        {
14.            for (int i = increment; i < sequence.length; i++)
15.            {
16.                int j = i;
17.                int temp = sequence[i];
18.                while (j >= increment && sequence[j - increment] > temp)
19.                {
20.                    sequence[j] = sequence[j - increment];
21.                    j = j - increment;
22.                }
23.                sequence[j] = temp;
24.            }
25.            if (increment == 2)
26.                increment = 1;
27.            else
28.                increment *= (5.0 / 11);
29.
30.        }
31.    }
32.
33.    static void printSequence(int[] sorted_sequence)
34.    {
35.        for (int i = 0; i < sorted_sequence.length; i++)
36.            System.out.print(sorted_sequence[i] + " ");
37.    }
38.
39.    public static void main(String args[])
40.    {
41.        System.out
42.            .println("Sorting of randomly generated numbers using SHELL SORT");
43.        Random random = new Random();
44.
45.        for (int i = 0; i < N; i++)
46.            sequence[i] = Math.abs(random.nextInt(100));
47.
48.        System.out.println("\nOriginal Sequence: ");
49.        printSequence(sequence);
50.
51.        System.out.println("\nSorted Sequence: ");
52.        shellSort();
53.        printSequence(sequence);
54.    }
55.}
```

Output:

advertisement

```
$ javac Shell_Sort.java  
$ java Shell_Sort
```

Sorting of randomly generated numbers using BUBBLE SORT

Original Sequence:

67 57 55 13 83 80 29 89 30 46 68 71 6 12 5 3 68 8 18 6

Sorted Sequence:

3 5 6 6 8 12 13 18 29 30 46 55 57 67 68 68 71 80 83 89

239.Java Program to Perform Sorting Using B-Tree

[« Prev](#)

[Next »](#)

This is a java program to implement sorting using B-Trees. Binary Search Trees are the type of B trees that self organizes. Inorder traversal of BSTs results in the sorted sequence of elements in the tree.

Here is the source code of the Java Program to Perform Sorting Using B-Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform sorting using BTree, Inorder traversal of BST results in sorting of numbers
2. import java.util.Random;
3.
4. /* Class BSTNode */
5. class BSTNodes {
6.     BSTNodes left, right;
7.     int data;
8.
9.     /* Constructor */
10.    public BSTNodes()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    /* Constructor */
18.    public BSTNodes(int n)
19.    {
20.        left = null;
21.        right = null;
22.        data = n;
23.    }
24.
25.    /* Function to set left node */
26.    public void setLeft(BSTNodes n)
27.    {
28.        left = n;
29.    }
30.
31.    /* Function to set right node */
32.    public void setRight(BSTNodes n)
33.    {
34.        right = n;
35.    }
36.
37.    /* Function to get left node */
38.    public BSTNodes getLeft()
39.    {
40.        return left;
41.    }
42.
43.    /* Function to get right node */
44.    public BSTNodes getRight()
45.    {
46.        return right;
47.    }
48.
49.    /* Function to set data to node */
50.    public void setData(int d)
51.    {
52.        data = d;
53.    }
54.
```

```

55. /* Function to get data from node */
56. public int getData()
57. {
58.     return data;
59. }
60.}
61.
62./* Class BST */
63.class BT {
64.    private BSTNode root;
65.
66. /* Constructor */
67. public BT()
68. {
69.     root = null;
70. }
71.
72. /* Function to check if tree is empty */
73. public boolean isEmpty()
74. {
75.     return root == null;
76. }
77.
78. /* Functions to insert data */
79. public void insert(int data)
80. {
81.     root = insert(root, data);
82. }
83.
84. /* Function to insert data recursively */
85. private BSTNode insert(BSTNode node, int data)
86. {
87.     if (node == null)
88.         node = new BSTNode(data);
89.     else
90.     {
91.         if (data <= node.getData())
92.             node.left = insert(node.left, data);
93.         else
94.             node.right = insert(node.right, data);
95.     }
96.     return node;
97. }
98.
99. /* Functions to delete data */
100. public void delete(int k)
101. {
102.     if (isEmpty())
103.         System.out.println("Tree Empty");
104.     else if (search(k) == false)
105.         System.out.println("Sorry " + k + " is not present");
106.     else
107.     {
108.         root = delete(root, k);
109.         System.out.println(k + " deleted from the tree");
110.     }
111. }
112.
113. private BSTNode delete(BSTNode root, int k)
114. {
115.     BSTNode p, p2, n;
116.     if (root.getData() == k)
117.     {
118.         BSTNode lt, rt;
119.         lt = root.getLeft();
120.         rt = root.getRight();

```

```

121.     if (lt == null && rt == null)
122.         return null;
123.     else if (lt == null)
124.     {
125.         p = rt;
126.         return p;
127.     }
128.     else if (rt == null)
129.     {
130.         p = lt;
131.         return p;
132.     }
133.     else
134.     {
135.         p2 = rt;
136.         p = rt;
137.         while (p.getLeft() != null)
138.             p = p.getLeft();
139.         p.setLeft(lt);
140.         return p2;
141.     }
142. }
143. if (k < root.getData())
144. {
145.     n = delete(root.getLeft(), k);
146.     root.setLeft(n);
147. }
148. else
149. {
150.     n = delete(root.getRight(), k);
151.     root.setRight(n);
152. }
153. return root;
154. }
155.
156. /* Functions to count number of nodes */
157. public int countNodes()
158. {
159.     return countNodes(root);
160. }
161.
162. /* Function to count number of nodes recursively */
163. private int countNodes(BSTNode r)
164. {
165.     if (r == null)
166.         return 0;
167.     else
168.     {
169.         int l = 1;
170.         l += countNodes(r.getLeft());
171.         l += countNodes(r.getRight());
172.         return l;
173.     }
174. }
175.
176. /* Functions to search for an element */
177. public boolean search(int val)
178. {
179.     return search(root, val);
180. }
181.
182. /* Function to search for an element recursively */
183. private boolean search(BSTNode r, int val)
184. {
185.     boolean found = false;
186.     while ((r != null) && !found)

```

```

187.     {
188.         int rval = r.getData();
189.         if (val < rval)
190.             r = r.getLeft();
191.         else if (val > rval)
192.             r = r.getRight();
193.         else
194.             {
195.                 found = true;
196.                 break;
197.             }
198.         found = search(r, val);
199.     }
200.     return found;
201. }
202.
203. /* Function for inorder traversal */
204. public void inorder()
205. {
206.     inorder(root);
207. }
208.
209. private void inorder(BSTNode r)
210. {
211.     if (r != null)
212.     {
213.         inorder(r.getLeft());
214.         System.out.print(r.getData() + " ");
215.         inorder(r.getRight());
216.     }
217. }
218.
219. /* Function for preorder traversal */
220. public void preorder()
221. {
222.     preorder(root);
223. }
224.
225. private void preorder(BSTNode r)
226. {
227.     if (r != null)
228.     {
229.         System.out.print(r.getData() + " ");
230.         preorder(r.getLeft());
231.         preorder(r.getRight());
232.     }
233. }
234.
235. /* Function for postorder traversal */
236. public void postorder()
237. {
238.     postorder(root);
239. }
240.
241. private void postorder(BSTNode r)
242. {
243.     if (r != null)
244.     {
245.         postorder(r.getLeft());
246.         postorder(r.getRight());
247.         System.out.print(r.getData() + " ");
248.     }
249. }
250. }
251.
252. public class Sort_BTree

```

```
253. {
254.     public static int N = 20;
255.
256.     public static void main(String args[])
257.     {
258.         Random random = new Random();
259.         BT bt = new BT();
260.
261.         System.out
262.             .println("Sorting of randomly generated numbers using B TREE");
263.
264.         for (int i = 0; i < N; i++)
265.             bt.insert(Math.abs(random.nextInt(100)));
266.
267.         System.out.println("The elements of the tree: ");
268.         bt.preorder();
269.
270.         System.out.println("\nThe sorted sequence is: ");
271.         bt.inorder();
272.     }
273. }
```

Output:

advertisement

```
$ javac Sort_BTTree.java
$ java Sort_BTTree
```

```
Sorting of randomly generated numbers using B TREE
The elements of the tree:
45 23 3 4 26 39 32 30 32 83 64 50 47 54 64 67 75 88 94 95
The sorted sequence is:
3 4 23 26 30 32 32 39 45 47 50 54 64 64 67 75 83 88 94 95
```

240.Java Program to Create a Balanced Binary Tree of the Incoming Data

[« Prev](#)

[Next »](#)

This is a Java Program to implement Self Balancing Binary Tree. A self-balancing (or height-balanced) binary tree is any node-based binary tree that automatically keeps its height (maximal number of levels below the root) small in the face of arbitrary item insertions and deletions. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets. The implementation of self balancing binary tree is similar to that of a AVL Tree data structure.

Here is the source code of the Java Program to Create a Balanced Binary Tree of the Incoming Data. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to make a self balancing binary tree for the input data
2. import java.util.Scanner;
3.
4. class SBBSTNodes
5. {
6.     SBBSTNodes left, right;
7.     int data;
8.     int height;
9.
10.    public SBBSTNodes()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.        height = 0;
16.    }
17.
18.    public SBBSTNodes(int n)
19.    {
20.
21.        left = null;
22.        right = null;
23.        data = n;
24.        height = 0;
25.    }
26.}
27.
28.class SelfBalancingBinarySearchTrees
29.
30.    private SBBSTNodes root;
31.
32.    public SelfBalancingBinarySearchTrees()
33.    {
34.        root = null;
35.    }
36.
37.    public boolean isEmpty()
38.    {
39.        return root == null;
40.    }
41.
42.    public void clear()
43.    {
44.        root = null;
45.    }
46.
47.    public void insert(int data)
```

```

48. {
49.     root = insert(data, root);
50. }
51.
52. private int height(SBBSTNodes t)
53. {
54.
55.     return t == null ? -1 : t.height;
56. }
57.
58. private int max(int lhs, int rhs)
59. {
60.     return lhs > rhs ? lhs : rhs;
61. }
62.
63. private SBBSTNodes insert(int x, SBBSTNodes t)
64. {
65.     if (t == null)
66.         t = new SBBSTNodes(x);
67.     else if (x < t.data)
68.     {
69.         t.left = insert(x, t.left);
70.         if (height(t.left) - height(t.right) == 2)
71.             if (x < t.left.data)
72.                 t = rotateWithLeftChild(t);
73.             else
74.                 t = doubleWithLeftChild(t);
75.     } else if (x > t.data)
76.     {
77.         t.right = insert(x, t.right);
78.         if (height(t.right) - height(t.left) == 2)
79.             if (x > t.right.data)
80.                 t = rotateWithRightChild(t);
81.             else
82.                 t = doubleWithRightChild(t);
83.     } else
84.     ;
85.     t.height = max(height(t.left), height(t.right)) + 1;
86.     return t;
87. }
88.
89. private SBBSTNodes rotateWithLeftChild(SBBSTNodes k2)
90. {
91.     SBBSTNodes k1 = k2.left;
92.     k2.left = k1.right;
93.     k1.right = k2;
94.     k2.height = max(height(k2.left), height(k2.right)) + 1;
95.     k1.height = max(height(k1.left), k2.height) + 1;
96.     return k1;
97. }
98.
99. private SBBSTNodes rotateWithRightChild(SBBSTNodes k1)
100. {
101.     SBBSTNodes k2 = k1.right;
102.     k1.right = k2.left;
103.     k2.left = k1;
104.     k1.height = max(height(k1.left), height(k1.right)) + 1;
105.     k2.height = max(height(k2.right), k1.height) + 1;
106.     return k2;
107. }
108.
109. private SBBSTNodes doubleWithLeftChild(SBBSTNodes k3)
110. {
111.     k3.left = rotateWithRightChild(k3.left);
112.     return rotateWithLeftChild(k3);
113. }

```

```

114.
115.     private SBBSTNodes doubleWithRightChild(SBBSTNodes k1)
116.     {
117.         k1.right = rotateWithLeftChild(k1.right);
118.         return rotateWithRightChild(k1);
119.     }
120.
121.     public int countNodes()
122.     {
123.         return countNodes(root);
124.     }
125.
126.     private int countNodes(SBBSTNodes r)
127.     {
128.         if (r == null)
129.             return 0;
130.         else
131.         {
132.             int l = 1;
133.             l += countNodes(r.left);
134.             l += countNodes(r.right);
135.             return l;
136.         }
137.     }
138.
139.     public boolean search(int val)
140.     {
141.         return search(root, val);
142.     }
143.
144.     private boolean search(SBBSTNodes r, int val)
145.     {
146.         boolean found = false;
147.         while ((r != null) && !found)
148.         {
149.             int rval = r.data;
150.             if (val < rval)
151.                 r = r.left;
152.             else if (val > rval)
153.                 r = r.right;
154.             else
155.             {
156.                 found = true;
157.                 break;
158.             }
159.             found = search(r, val);
160.         }
161.         return found;
162.     }
163.
164.     public void inorder()
165.     {
166.         inorder(root);
167.     }
168.
169.     private void inorder(SBBSTNodes r)
170.     {
171.         if (r != null)
172.         {
173.             inorder(r.left);
174.             System.out.print(r.data + " ");
175.             inorder(r.right);
176.         }
177.     }
178.
179.     public void preorder()

```

```

180. {
181.     preorder(root);
182. }
183.
184.
185. private void preorder(SBBSTNodes r)
186. {
187.     if (r != null)
188.     {
189.         System.out.print(r.data + " ");
190.         preorder(r.left);
191.         preorder(r.right);
192.     }
193. }
194.
195. public void postorder()
196. {
197.     postorder(root);
198. }
199.
200. private void postorder(SBBSTNodes r)
201. {
202.     if (r != null)
203.     {
204.         postorder(r.left);
205.         postorder(r.right);
206.         System.out.print(r.data + " ");
207.     }
208. }
209. }
210.
211.public class Balanced_B_Tree
212. {
213.     public static void main(String[] args)
214.     {
215.         Scanner scan = new Scanner(System.in);
216.
217.         SelfBalancingBinarySearchTrees sbbst = new SelfBalancingBinarySearchTrees();
218.         System.out.println("Self Balancing Tree\n");
219.
220.         int N = 10;
221.         for (int i = 0; i < N; i++)
222.             sbbst.insert(scan.nextInt());
223.
224.         System.out.println("\nPre-order :");
225.         sbbst.preorder();
226.         System.out.println("\nIn-order :");
227.         sbbst.inorder();
228.         System.out.println("\nPost-order :");
229.         sbbst.postorder();
230.         scan.close();
231.     }
232. }

```

Output:

advertisement

```
$ javac Balanced_B_Tree.java
$ java Balanced_B_Tree
```

Self Balancing Tree

```
45
46
48
98
```

23

34

65

59

21

10

Pre-order :

46 34 21 10 23 45 65 48 59 98

In-order :

10 21 23 34 45 46 48 59 65 98

Post-order :

10 23 21 45 34 59 48 98 65 46

241.Java Program to Print only Odd Numbered Levels of a Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement a Binary Tree and print the level order traversal of the same, such that only odd levels are considered. At current level check nodes in next to next level and put them in Queue, dequeue them one by one and print them.

Here is the source code of the Java Program to Print only Odd Numbered Levels of a Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to print only odd levels of the given tree
2. import java.util.Queue;
3. import java.util.LinkedList;
4.
5. public class BinaryTreePrintOddLevels
6. {
7.     private static class Node<T>
8.     {
9.         public Node<T> left;
10.        public Node<T> right;
11.        public T data;
12.
13.        public Node(T data)
14.        {
15.            this.data = data;
16.        }
17.
18.        public Node<T> getLeft()
19.        {
20.            return this.left;
21.        }
22.
23.        public void setLeft(Node<T> left)
24.        {
25.            this.left = left;
26.        }
27.
28.        public Node<T> getRight()
29.        {
30.            return this.right;
31.        }
32.
33.        public void setRight(Node<T> right)
34.        {
35.            this.right = right;
36.        }
37.    }
38.
39.    public static void preorder(Node<?> n)
40.    {
41.        if (n != null)
42.        {
43.            System.out.print(n.data + " ");
44.            preorder(n.getLeft());
45.            preorder(n.getRight());
46.        }
47.    }
48.
49.    public static void inorder(Node<?> n)
50.    {
```

```

51.     if (n != null)
52.     {
53.         inorder(n.getLeft());
54.         System.out.print(n.data + " ");
55.         inorder(n.getRight());
56.     }
57. }
58.
59. public static void postorder(Node<?> n)
60. {
61.     if (n != null)
62.     {
63.         postorder(n.getLeft());
64.         postorder(n.getRight());
65.         System.out.print(n.data + " ");
66.     }
67. }
68.
69. public static void levelorder(Node<?> n)
70. {
71.     Queue<Node<?>> nodequeue = new LinkedList<Node<?>>();
72.     if (n != null)
73.         nodequeue.add(n);
74.     while (!nodequeue.isEmpty())
75.     {
76.         Node<?> next = nodequeue.remove();
77.         System.out.print(next.data + " ");
78.         if (next.getLeft() != null)
79.         {
80.             if (next.getLeft().getLeft() != null)
81.                 nodequeue.add(next.getLeft().getLeft());
82.             if (next.getLeft().getRight() != null)
83.                 nodequeue.add(next.getLeft().getRight());
84.         }
85.         if (next.getRight() != null)
86.         {
87.             if (next.getRight().getLeft() != null)
88.                 nodequeue.add(next.getRight().getLeft());
89.             if (next.getRight().getRight() != null)
90.                 nodequeue.add(next.getRight().getRight());
91.         }
92.     }
93. }
94.
95. public static void main(final String[] args)
96. {
97.     Node<Integer> one = new Node<Integer>(1);
98.     Node<Integer> two = new Node<Integer>(2);
99.     Node<Integer> three = new Node<Integer>(3);
100.    Node<Integer> four = new Node<Integer>(4);
101.    Node<Integer> five = new Node<Integer>(5);
102.    Node<Integer> six = new Node<Integer>(6);
103.    Node<Integer> seven = new Node<Integer>(7);
104.    Node<Integer> eight = new Node<Integer>(8);
105.    Node<Integer> nine = new Node<Integer>(9);
106.
107.    one.setLeft(two);
108.    one.setRight(three);
109.    two.setLeft(four);
110.    two.setRight(five);
111.    three.setLeft(six);
112.    four.setLeft(seven);
113.    six.setLeft(eight);
114.    six.setRight(nine);
115.
116.    System.out.println("\nPre-Order of the Tree");

```

```
117.    preorder(one);
118.    System.out.println("\nIn-Order of the Tree");
119.    inorder(one);
120.    System.out.println("\nPost-Order of the Tree");
121.    postorder(one);
122.    System.out.println("\nLevel-Order Odd Levels of the Tree");
123.    levelorder(one);
124.
125. }
126.}
```

Output:

advertisement

```
$ javac BinaryTreePrintOddLevels.java
$ java BinaryTreePrintOddLevels
```

```
Pre-Order of the Tree
1 2 4 7 5 3 6 8 9
In-Order of the Tree
7 4 2 5 1 8 6 9 3
Post-Order of the Tree
7 4 5 2 8 9 6 3 1
Level-Order Odd Levels of the Tree
1 4 5 6
```

242.Java Program to Check the Connectivity of Graph Using BFS

[« Prev](#)

[Next »](#)

This is a Java Program to implement graph and check the connectivity between nodes using a standard Breadth First Search algorithm. Algorithm visits the node that was traversed first or appeared in liked list representation of the node or first come first serve basis. We create a visited array to avoid revisiting a node. If destination node appears in visited array, source and destination nodes are connected, not otherwise.

Here is the source code of the Java Program to Check the Connectivity of Graph Using BFS. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check the nodes are connected using BFS
2. import java.util.LinkedList;
3. import java.util.Queue;
4. import java.util.Scanner;
5.
6. public class Connectivity_BFS
7. {
8.     private final int    vertices;
9.     private int[][]    adjacency_matrix;
10.    private Queue<Integer> queue;
11.
12.    public Connectivity_BFS(int v)
13.    {
14.        vertices = v;
15.        adjacency_matrix = new int[vertices + 1][vertices + 1];
16.        queue = new LinkedList<Integer>();
17.    }
18.
19.    public void makeEdge(int to, int from, int edge)
20.    {
21.        try
22.        {
23.            adjacency_matrix[to][from] = edge;
24.            adjacency_matrix[from][to] = edge;
25.        } catch (ArrayIndexOutOfBoundsException index)
26.        {
27.            System.out.println("The vertices does not exists");
28.        }
29.    }
30.
31.    public int getEdge(int to, int from)
32.    {
33.        try
34.        {
35.            return adjacency_matrix[to][from];
36.        } catch (ArrayIndexOutOfBoundsException index)
37.        {
38.            System.out.println("The vertices does not exists");
39.        }
40.        return -1;
41.    }
42.
43.    public void bfs(int source)
44.    {
45.        int number_of_nodes = adjacency_matrix[source].length - 1;
46.        int[] visited = new int[number_of_nodes + 1];
47.        int i, element;
48.        visited[source] = 1;
49.        queue.add(source);
```

```

50.    while (!queue.isEmpty())
51.    {
52.        element = queue.remove();
53.        i = 1;// element;
54.        while (i <= number_of_nodes)
55.        {
56.            if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
57.            {
58.                queue.add(i);
59.                visited[i] = 1;
60.            }
61.            i++;
62.        }
63.    }
64.
65.    System.out.print("The source node " + source + " is connected to: ");
66.    int count = 0;
67.    for (int v = 1; v <= number_of_nodes; v++)
68.    {
69.        if (visited[v] == 1)
70.        {
71.            System.out.print(v + " ");
72.            count++;
73.        }
74.    }
75.    if (count == number_of_nodes)
76.        System.out.print("\nThe Graph is Connected ");
77.    else
78.        System.out.print("\nThe Graph is Disconnected ");
79.    }
80. public static void main(String args[])
81. {
82.     int v, e, count = 0, to = 0, from = 0;
83.     Scanner sc = new Scanner(System.in);
84.     Connectivity_BFS graph;
85.     System.out.println("The Undirected Graph Connectivity Test");
86.     try
87.     {
88.         System.out.println("Enter the number of vertices: ");
89.         v = sc.nextInt();
90.         System.out.println("Enter the number of edges: ");
91.         e = sc.nextInt();
92.
93.         graph = new Connectivity_BFS(v);
94.
95.         System.out.println("Enter the edges: <to> <from>");
96.         while (count <= e)
97.         {
98.             to = sc.nextInt();
99.             from = sc.nextInt();
100.
101.            graph.makeEdge(to, from, 1);
102.            count++;
103.        }
104.
105.        System.out.println("The adjacency matrix for the given graph is: ");
106.        System.out.print(" ");
107.        for (int i = 1; i <= v; i++)
108.        {
109.            System.out.print(i + " ");
110.            System.out.println();
111.        }
112.        for (int i = 1; i <= v; i++)
113.        {
114.            System.out.print(i + " ");
115.            for (int j = 1; j <= v; j++)
116.                System.out.print(graph.getEdge(i, j) + " ");

```

```
116.         System.out.println();
117.     }
118.
119.     System.out.println("Enter the Source Node: ");
120.     int sourceNode = sc.nextInt();
121.     graph.bfs(sourceNode);
122.
123. } catch (Exception E)
124. {
125.     System.out.println("Something went wrong");
126. }
127.
128. sc.close();
129. }
130. }
```

Output:

advertisement

```
$ javac Connectivity_BFS.java
$ java Connectivity_BFS
```

The Undirected Graph Connectivity Test(BFS)

Enter the number of vertices:

4

Enter the number of edges:

2

Enter the edges: <to> <from>

1 2

3 4

The adjacency matrix for the given graph is:

1	2	3	4
1	0	1	0
2	1	0	0
3	0	0	0
4	0	0	1

Enter the Source Node:

3

The source node 3 is connected to: 3 4

The Graph is Disconnected

The Undirected Graph Connectivity Test(BFS)

Enter the number of vertices:

4

Enter the number of edges:

5

Enter the edges: <to> <from>

1 2

2 3

3 4

1 4

1 3

The adjacency matrix for the given graph is:

1	2	3	4
1	0	1	1
2	1	0	1
3	1	1	0
4	1	0	1

Enter the Source Node:

4

The source node 4 is connected to: 1 2 3 4

The Graph is Connected

243.Java Program to Check the Connectivity of Graph Using DFS

[« Prev](#)

[Next »](#)

This is a Java Program to implement graph and check the connectivity between nodes using a standard Depth First Search algorithm. Algorithm visits the node that was traversed last or last come first serve basis. We create a visited array to avoid revisiting a node. If destination node appears in visited array, source and destination nodes are connected, not otherwise.

Here is the source code of the Java Program to Check the Connectivity of Graph Using DFS. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check the connectivity of a graph using DFS
2. import java.util.Scanner;
3. import java.util.Stack;
4.
5. public class Connectivity_DFS
6. {
7.     private final int    vertices;
8.     private int[][]    adjacency_matrix;
9.     private Stack<Integer> stack;
10.
11.    public Connectivity_DFS(int v)
12.    {
13.        vertices = v;
14.        adjacency_matrix = new int[vertices + 1][vertices + 1];
15.        stack = new Stack<Integer>();
16.    }
17.
18.    public void makeEdge(int to, int from, int edge)
19.    {
20.        try
21.        {
22.            adjacency_matrix[to][from] = edge;
23.            adjacency_matrix[from][to] = edge;
24.        } catch (ArrayIndexOutOfBoundsException index)
25.        {
26.            System.out.println("The vertices does not exists");
27.        }
28.    }
29.
30.    public int getEdge(int to, int from)
31.    {
32.        try
33.        {
34.            return adjacency_matrix[to][from];
35.        } catch (ArrayIndexOutOfBoundsException index)
36.        {
37.            System.out.println("The vertices does not exists");
38.        }
39.        return -1;
40.    }
41.
42.    public void dfs(int source)
43.    {
44.        int number_of_nodes = adjacency_matrix[source].length - 1;
45.        int[] visited = new int[number_of_nodes + 1];
46.        int i, element;
47.        visited[source] = 1;
48.        stack.push(source);
49.        while (!stack.isEmpty())
```

```

50. {
51.     element = stack.pop();
52.     i = 1; // element;
53.     while (i <= number_of_nodes)
54.     {
55.         if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
56.         {
57.             stack.push(i);
58.             visited[i] = 1;
59.         }
60.         i++;
61.     }
62. }
63.
64. System.out.print("The source node " + source + " is connected to: ");
65. int count = 0;
66. for (int v = 1; v <= number_of_nodes; v++)
67. {
68.     if (visited[v] == 1)
69.     {
70.         System.out.print(v + " ");
71.         count++;
72.     }
73. }
74. if (count == number_of_nodes)
75.     System.out.print("\nThe Graph is Connected ");
76. else
77.     System.out.print("\nThe Graph is Disconnected ");
78.
79. public static void main(String args[])
80. {
81.     int v, e, count = 1, to = 0, from = 0;
82.     Scanner sc = new Scanner(System.in);
83.     Connectivity_DFS graph;
84.     System.out.println("The Undirected Graph Connectivity Test");
85.     try
86.     {
87.         System.out.println("Enter the number of vertices: ");
88.         v = sc.nextInt();
89.         System.out.println("Enter the number of edges: ");
90.         e = sc.nextInt();
91.
92.         graph = new Connectivity_DFS(v);
93.
94.         System.out.println("Enter the edges: <to> <from>");
95.         while (count <= e)
96.         {
97.             to = sc.nextInt();
98.             from = sc.nextInt();
99.
100.            graph.makeEdge(to, from, 1);
101.            count++;
102.        }
103.
104.        System.out.println("The adjacency matrix for the given graph is: ");
105.        System.out.print(" ");
106.        for (int i = 1; i <= v; i++)
107.            System.out.print(i + " ");
108.        System.out.println();
109.
110.        for (int i = 1; i <= v; i++)
111.        {
112.            System.out.print(i + " ");
113.            for (int j = 1; j <= v; j++)
114.                System.out.print(graph.getEdge(i, j) + " ");
115.            System.out.println();

```

```
116.    }
117.
118.    System.out.println("Enter the Source Node: ");
119.    int sourceNode = sc.nextInt();
120.    graph.dfs(sourceNode);
121.
122. } catch (Exception E)
123. {
124.     System.out.println("Somthing went wrong");
125. }
126.
127. sc.close();
128.
129.}
```

Output:

advertisement

```
$ javac Connectivity_DFS.java
$ java Connectivity_DFS
```

The Undirected Graph Connectivity Test(DFS)

Enter the number of vertices:

4

Enter the number of edges:

2

Enter the edges: <to> <from>

1 2

1 3

The adjacency matrix for the given graph is:

```
1 2 3 4
1 0 1 1 0
2 1 0 0 0
3 1 0 0 0
4 0 0 0 0
```

Enter the Source Node:

2

The source node 2 is connected to: 1 2 3

The Graph is Disconnected

The Undirected Graph Connectivity Test(DFS)

Enter the number of vertices:

4

Enter the number of edges:

4

Enter the edges: <to> <from>

1 2

1 3

1 4

2 4

The adjacency matrix for the given graph is:

```
1 2 3 4
1 0 1 1 1
2 1 0 0 1
3 1 0 0 0
4 1 1 0 0
```

Enter the Source Node:

4

The source node 4 is connected to: 1 2 3 4

The Graph is Connected

244.Java Program to Perform Deletion in a BST

[« Prev](#)

[Next »](#)

This is a Java Program to perform deletion in the binary search tree.

Here is the source code of the Java Program to Perform Deletion in a BST. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to delete elements from Binary Search Tree
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. class BSTNode
6. {
7.     BSTNode left, right;
8.     int data;
9.
10.    public BSTNode()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BSTNode(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BSTNode n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BSTNode n)
30.    {
31.        right = n;
32.    }
33.
34.    public BSTNode getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BSTNode getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
48.
49.    public int getData()
50.    {
51.        return data;
52.    }
53.}
54.
55.class BSTree
```

```
56. {
57.     private BSTNode root;
58.
59.     public BSTree()
60.     {
61.         root = null;
62.     }
63.
64.     public boolean isEmpty()
65.     {
66.         return root == null;
67.     }
68.
69.     public void insert(int data)
70.     {
71.         root = insert(root, data);
72.     }
73.
74.     private BSTNode insert(BSTNode node, int data)
75.     {
76.         if (node == null)
77.             node = new BSTNode(data);
78.         else
79.         {
80.             if (data <= node.getData())
81.                 node.left = insert(node.left, data);
82.             else
83.                 node.right = insert(node.right, data);
84.         }
85.         return node;
86.     }
87.
88.     public void delete(int k)
89.     {
90.         if (isEmpty())
91.             System.out.println("Tree Empty");
92.         else if (search(k) == false)
93.             System.out.println("Sorry " + k + " is not present");
94.         else
95.         {
96.             root = delete(root, k);
97.             System.out.println(k + " deleted from the tree");
98.         }
99.     }
100.
101.    private BSTNode delete(BSTNode root, int k)
102.    {
103.        BSTNode p, p2, n;
104.        if (root.getData() == k)
105.        {
106.            BSTNode lt, rt;
107.            lt = root.getLeft();
108.            rt = root.getRight();
109.            if (lt == null && rt == null)
110.                return null;
111.            else if (lt == null)
112.            {
113.                p = rt;
114.                return p;
115.            } else if (rt == null)
116.            {
117.                p = lt;
118.                return p;
119.            } else
120.            {
121.                p2 = rt;
```

```

122.         p = rt;
123.         while (p.getLeft() != null)
124.             p = p.getLeft();
125.             p.setLeft(lt);
126.             return p2;
127.     }
128. }
129. if (k < root.getData())
130. {
131.     n = delete(root.getLeft(), k);
132.     root.setLeft(n);
133. } else
134. {
135.     n = delete(root.getRight(), k);
136.     root.setRight(n);
137. }
138. return root;
139. }
140.
141. public boolean search(int val)
142. {
143.     return search(root, val);
144. }
145.
146. private boolean search(BSTNode r, int val)
147. {
148.     boolean found = false;
149.     while ((r != null) && !found)
150.     {
151.         int rval = r.getData();
152.         if (val < rval)
153.             r = r.getLeft();
154.         else if (val > rval)
155.             r = r.getRight();
156.         else
157.         {
158.             found = true;
159.             break;
160.         }
161.         found = search(r, val);
162.     }
163.     return found;
164. }
165.
166. public void inorder()
167. {
168.     inorder(root);
169. }
170.
171. private void inorder(BSTNode r)
172. {
173.     if (r != null)
174.     {
175.         inorder(r.getLeft());
176.         System.out.print(r.getData() + " ");
177.         inorder(r.getRight());
178.     }
179. }
180.
181. public void preorder()
182. {
183.     preorder(root);
184. }
185.
186. private void preorder(BSTNode r)
187. {

```

```

188.     if (r != null)
189.     {
190.         System.out.print(r.getData() + " ");
191.         preorder(r.getLeft());
192.         preorder(r.getRight());
193.     }
194. }
195.
196. public void postorder()
197. {
198.     postorder(root);
199. }
200.
201. private void postorder(BSTNode r)
202. {
203.     if (r != null)
204.     {
205.         postorder(r.getLeft());
206.         postorder(r.getRight());
207.         System.out.print(r.getData() + " ");
208.     }
209. }
210. }
211.
212. public class Deletion_BST
213. {
214.     public static void main(String[] args)
215.     {
216.         BSTree bst = new BSTree();
217.         System.out.println("Binary Search Tree Deletion Test\n");
218.
219.         Scanner sc = new Scanner(System.in);
220.         Random random = new Random();
221.         int n = 15;
222.         for (int i = 0; i < n; i++)
223.             bst.insert(Math.abs(random.nextInt(100)));
224.         char ch;
225.         do
226.         {
227.             System.out.print("\nPost order : ");
228.             bst.postorder();
229.             System.out.print("\nPre order : ");
230.             bst.preorder();
231.             System.out.print("\nIn order : ");
232.             bst.inorder();
233.
234.             System.out.println("Enter integer element to delete");
235.             bst.delete(sc.nextInt());
236.             System.out.println("Continue deleting? <y>/<n> ");
237.             ch = sc.next().charAt(0);
238.         } while (ch == 'y' || ch == 'Y');
239.         sc.close();
240.     }
241. }

```

Output:

advertisement

```
$ javac Deletion_BST.java
$ java Deletion_BST
```

Binary Search Tree Deletion Test

```
Post order : 17 11 22 26 24 41 46 52 54 42 86 85 74 72 23
Pre order : 23 22 11 17 72 42 41 24 26 54 52 46 74 85 86
```

In order : 11 17 22 23 24 26 41 42 46 52 54 72 74 85 86

Enter integer element to delete

17

17 deleted from the tree

Continue deleting? <y>/<n>

y

Post order : 11 22 26 24 41 46 52 54 42 86 85 74 72 23

Pre order : 23 22 11 72 42 41 24 26 54 52 46 74 85 86

In order : 11 22 23 24 26 41 42 46 52 54 72 74 85 86

Enter integer element to delete

23

23 deleted from the tree

Continue deleting? <y>/<n>

y

Post order : 11 22 26 24 41 46 52 54 42 86 85 74 72

Pre order : 72 42 41 24 22 11 26 54 52 46 74 85 86

In order : 11 22 24 26 41 42 46 52 54 72 74 85 86

Enter integer element to delete

11

11 deleted from the tree

Continue deleting? <y>/<n>

n

245.Java Program to Perform Insertion in a BST

[« Prev](#)

[Next »](#)

This is a Java Program to perform insertion in the binary search tree.

Here is the source code of the Java Program to Perform Insertion in a BST. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to insert elements in a Binary Search Tree
2. import java.util.Scanner;
3.
4. class BSTNode
5. {
6.     BSTNode left, right;
7.     int data;
8.
9.     public BSTNode()
10.    {
11.        left = null;
12.        right = null;
13.        data = 0;
14.    }
15.
16.     public BSTNode(int n)
17.    {
18.        left = null;
19.        right = null;
20.        data = n;
21.    }
22.
23.     public void setLeft(BSTNode n)
24.    {
25.        left = n;
26.    }
27.
28.     public void setRight(BSTNode n)
29.    {
30.        right = n;
31.    }
32.
33.     public BSTNode getLeft()
34.    {
35.        return left;
36.    }
37.
38.     public BSTNode getRight()
39.    {
40.        return right;
41.    }
42.
43.     public void setData(int d)
44.    {
45.        data = d;
46.    }
47.
48.     public int getData()
49.    {
50.        return data;
51.    }
52.}
53.
54.class BSTree
55.{
```

```
56. private BSTNode root;
57.
58. public BSTree()
59. {
60.     root = null;
61. }
62.
63. public boolean isEmpty()
64. {
65.     return root == null;
66. }
67.
68. public void insert(int data)
69. {
70.     root = insert(root, data);
71. }
72.
73. private BSTNode insert(BSTNode node, int data)
74. {
75.     if (node == null)
76.         node = new BSTNode(data);
77.     else
78.     {
79.         if (data <= node.getData())
80.             node.left = insert(node.left, data);
81.         else
82.             node.right = insert(node.right, data);
83.     }
84.     return node;
85. }
86.
87. public void inorder()
88. {
89.     inorder(root);
90. }
91.
92. private void inorder(BSTNode r)
93. {
94.     if (r != null)
95.     {
96.         inorder(r.getLeft());
97.         System.out.print(r.getData() + " ");
98.         inorder(r.getRight());
99.     }
100. }
101.
102. public void preorder()
103. {
104.     preorder(root);
105. }
106.
107. private void preorder(BSTNode r)
108. {
109.     if (r != null)
110.     {
111.         System.out.print(r.getData() + " ");
112.         preorder(r.getLeft());
113.         preorder(r.getRight());
114.     }
115. }
116.
117. public void postorder()
118. {
119.     postorder(root);
120. }
121.
```

```

122. private void postorder(BSTNode r)
123. {
124.     if (r != null)
125.     {
126.         postorder(r.getLeft());
127.         postorder(r.getRight());
128.         System.out.print(r.getData() + " ");
129.     }
130. }
131.}
132.
133. public class Insertion_BST
134. {
135.     public static void main(String[] args)
136.     {
137.         Scanner scan = new Scanner(System.in);
138.         BSTree bst = new BSTree();
139.         System.out.println("Binary Search Tree Insertion Test\n");
140.         int N = 10;
141.         for (int i = 0; i < N; i++)
142.         {
143.             bst.insert(scan.nextInt());
144.             System.out.print("\nPost order : ");
145.             bst.postorder();
146.             System.out.print("\nPre order : ");
147.             bst.preorder();
148.             System.out.print("\nIn order : ");
149.             bst.inorder();
150.         }
151.         scan.close();
152.     }
153. }

```

Output:

advertisement

```
$ java Insertion_BST.java
$ java Insertion_BST
```

Binary Search Tree Insertion Test

Add element: 3

```
Post order : 3
Pre order : 3
In order : 3
```

Add element: 5

```
Post order : 5 3
Pre order : 3 5
In order : 3 5
```

Add element: 4

```
Post order : 4 5 3
Pre order : 3 5 4
In order : 3 4 5
```

Add element: 4

```
Post order : 4 4 5 3
Pre order : 3 5 4 4
In order : 3 4 4 5
```

Add element: 6

Post order : 4 4 6 5 3
Pre order : 3 5 4 4 6
In order : 3 4 4 5 6

Add element: 45

Post order : 4 4 45 6 5 3
Pre order : 3 5 4 4 6 45
In order : 3 4 4 5 6 45

Add element: 7578

Post order : 4 4 7578 45 6 5 3
Pre order : 3 5 4 4 6 45 7578
In order : 3 4 4 5 6 45 7578

Add element: 54651

Post order : 4 4 54651 7578 45 6 5 3
Pre order : 3 5 4 4 6 45 7578 54651
In order : 3 4 4 5 6 45 7578 54651

Add element: 22

Post order : 4 4 22 54651 7578 45 6 5 3
Pre order : 3 5 4 4 6 45 22 7578 54651
In order : 3 4 4 5 6 22 45 7578 54651

Add element: 34

Post order : 4 4 34 22 54651 7578 45 6 5 3
Pre order : 3 5 4 4 6 45 22 34 7578 54651
In order : 3 4 4 5 6 22 34 45 7578 54651

246.Java Program to Perform Search in a BST

[« Prev](#)

[Next »](#)

This is a Java Program to perform search an element in the binary search tree.

Here is the source code of the Java Program to Perform Search in a BST. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element in a Binary Search Tree
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. class BSTNode
6. {
7.     BSTNode left, right;
8.     int data;
9.
10.    public BSTNode()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BSTNode(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BSTNode n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BSTNode n)
30.    {
31.        right = n;
32.    }
33.
34.    public BSTNode getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BSTNode getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
48.
49.    public int getData()
50.    {
51.        return data;
52.    }
53.}
54.
55.class BSTree
```

```

56.
57.     private BSTNode root;
58.
59.     public BSTree()
60.     {
61.         root = null;
62.     }
63.
64.     public boolean isEmpty()
65.     {
66.         return root == null;
67.     }
68.
69.     public void insert(int data)
70.     {
71.         root = insert(root, data);
72.     }
73.
74.     private BSTNode insert(BSTNode node, int data)
75.     {
76.         if (node == null)
77.             node = new BSTNode(data);
78.         else
79.         {
80.             if (data <= node.getData())
81.                 node.left = insert(node.left, data);
82.             else
83.                 node.right = insert(node.right, data);
84.         }
85.         return node;
86.     }
87.
88.     public boolean search(int val)
89.     {
90.         return search(root, val);
91.     }
92.
93.     private boolean search(BSTNode r, int val)
94.     {
95.         boolean found = false;
96.         while ((r != null) && !found)
97.         {
98.             int rval = r.getData();
99.             if (val < rval)
100.                 r = r.getLeft();
101.             else if (val > rval)
102.                 r = r.getRight();
103.             else
104.             {
105.                 found = true;
106.                 break;
107.             }
108.             found = search(r, val);
109.         }
110.         return found;
111.     }
112.
113.     public void inorder()
114.     {
115.         inorder(root);
116.     }
117.
118.     private void inorder(BSTNode r)
119.     {
120.         if (r != null)
121.         {

```

```

122.     inorder(r.getLeft());
123.     System.out.print(r.getData() + " ");
124.     inorder(r.getRight());
125.   }
126. }
127.
128. public void preorder()
129. {
130.   preorder(root);
131. }
132.
133. private void preorder(BSTNode r)
134. {
135.   if (r != null)
136.   {
137.     System.out.print(r.getData() + " ");
138.     preorder(r.getLeft());
139.     preorder(r.getRight());
140.   }
141. }
142.
143. public void postorder()
144. {
145.   postorder(root);
146. }
147.
148. private void postorder(BSTNode r)
149. {
150.   if (r != null)
151.   {
152.     postorder(r.getLeft());
153.     postorder(r.getRight());
154.     System.out.print(r.getData() + " ");
155.   }
156. }
157. }
158.
159. public class Searching_Tree
160. {
161.   public static void main(String[] args)
162.   {
163.     BSTree bst = new BSTree();
164.     System.out.println("Binary Search Tree Deletion Test\n");
165.
166.     Scanner input = new Scanner(System.in);
167.     Random rand = new Random();
168.     int elements = 15;
169.     for (int i = 0; i < elements; i++)
170.       bst.insert(Math.abs(rand.nextInt(100)));
171.
172.     System.out.println("Enter integer element to search");
173.     System.out.println("Search result : " + bst.search(input.nextInt()));
174.
175.     System.out.print("\nPost order : ");
176.     bst.postorder();
177.     System.out.print("\nPre order : ");
178.     bst.preorder();
179.     System.out.print("\nIn order : ");
180.     bst.inorder();
181.
182.     input.close();
183.   }
184. }

```

Output:

advertisement

```
$ javac Searching_BST.java  
$ java Searching_BST
```

Binary Search Tree Searching Test

Enter integer element to search

15

Search result : true

Post order : 0 10 5 24 15 38 4 63 55 72 67 79 97 80 49

Pre order : 49 4 0 38 15 5 10 24 80 79 67 55 63 72 97

In order : 0 4 5 10 15 24 38 49 55 63 67 72 79 80 97

247.Java Program to Find the Shortest Path Between Two Vertices Using Dijkstra's Algorithm

[« Prev](#)

[Next »](#)

This is a Java Program to perform Dijkstra's Shortest path algorithm. For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

Here is the source code of the Java Program to Find the Shortest Path Between Two Vertices Using Dijkstra's Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the shortest path between source vertex and destination vertex
2. import java.util.HashSet;
3. import java.util.InputMismatchException;
4. import java.util.Iterator;
5. import java.util.Scanner;
6. import java.util.Set;
7.
8. public class Dijkstras_Shortest_Path
9. {
10.     private int      distances[];
11.     private Set<Integer> settled;
12.     private Set<Integer> unsettled;
13.     private int      number_of_nodes;
14.     private int      adjacencyMatrix[][];
15.
16.     public Dijkstras_Shortest_Path(int number_of_nodes)
17.     {
18.         this.number_of_nodes = number_of_nodes;
19.         distances = new int[number_of_nodes + 1];
20.         settled = new HashSet<Integer>();
21.         unsettled = new HashSet<Integer>();
22.         adjacencyMatrix = new int[number_of_nodes + 1][number_of_nodes + 1];
23.     }
24.
25.     public void dijkstra_algorithm(int adjacency_matrix[][], int source)
26.     {
27.         int evaluationNode;
28.         for (int i = 1; i <= number_of_nodes; i++)
29.             for (int j = 1; j <= number_of_nodes; j++)
30.                 adjacencyMatrix[i][j] = adjacency_matrix[i][j];
31.
32.         for (int i = 1; i <= number_of_nodes; i++)
33.         {
34.             distances[i] = Integer.MAX_VALUE;
35.         }
36.
37.         unsettled.add(source);
38.         distances[source] = 0;
39.         while (!unsettled.isEmpty())
40.         {
41.             evaluationNode = getNodeWithMinimumDistanceFromUnsettled();
42.             unsettled.remove(evaluationNode);
43.             settled.add(evaluationNode);
44.         }
45.     }
46.
47.     private int getNodeWithMinimumDistanceFromUnsettled()
48.     {
49.         int minDistance = Integer.MAX_VALUE;
50.         int evaluationNode = -1;
51.         for (int i : unsettled)
52.         {
53.             if (distances[i] < minDistance)
54.             {
55.                 minDistance = distances[i];
56.                 evaluationNode = i;
57.             }
58.         }
59.         return evaluationNode;
60.     }
61.
62.     public static void main(String[] args)
63.     {
64.         Scanner scanner = new Scanner(System.in);
65.         Dijkstras_Shortest_Path dijkstra = new Dijkstras_Shortest_Path(5);
66.         System.out.println("Enter the weight of edges");
67.         for (int i = 0; i < 5; i++)
68.         {
69.             for (int j = 0; j < 5; j++)
70.             {
71.                 int weight = scanner.nextInt();
72.                 if (weight != 0)
73.                     dijkstra.adjacencyMatrix[i][j] = weight;
74.             }
75.         }
76.         System.out.println("Enter the source vertex");
77.         int source = scanner.nextInt();
78.         System.out.println("Enter the destination vertex");
79.         int destination = scanner.nextInt();
80.         dijkstra.dijkstra_algorithm(dijkstra.adjacencyMatrix, source);
81.         System.out.println("Shortest distance from " + source + " to " + destination + " is " + dijkstra.distances[destination]);
82.     }
83. }
```

```

44.     evaluateNeighbours(evaluationNode);
45.   }
46. }
47.
48. private int getNodeWithMinimumDistanceFromUnsettled()
49. {
50.     int min;
51.     int node = 0;
52.
53.     Iterator<Integer> iterator = unsettled.iterator();
54.     node = iterator.next();
55.     min = distances[node];
56.     for (int i = 1; i <= distances.length; i++)
57.     {
58.         if (unsettled.contains(i))
59.         {
60.             if (distances[i] <= min)
61.             {
62.                 min = distances[i];
63.                 node = i;
64.             }
65.         }
66.     }
67.     return node;
68. }
69.
70. private void evaluateNeighbours(int evaluationNode)
71. {
72.     int edgeDistance = -1;
73.     int newDistance = -1;
74.
75.     for (int destinationNode = 1; destinationNode <= number_of_nodes; destinationNode++)
76.     {
77.         if (!settled.contains(destinationNode))
78.         {
79.             if (adjacencyMatrix[evaluationNode][destinationNode] != Integer.MAX_VALUE)
80.             {
81.                 edgeDistance = adjacencyMatrix[evaluationNode][destinationNode];
82.                 newDistance = distances[evaluationNode] + edgeDistance;
83.                 if (newDistance < distances[destinationNode])
84.                 {
85.                     distances[destinationNode] = newDistance;
86.                 }
87.                 unsettled.add(destinationNode);
88.             }
89.         }
90.     }
91. }
92.
93. public static void main(String... arg)
94. {
95.     int adjacency_matrix[][];
96.     int number_of_vertices;
97.     int source = 0, destination = 0;
98.     Scanner scan = new Scanner(System.in);
99.     try
100.    {
101.        System.out.println("Enter the number of vertices");
102.        number_of_vertices = scan.nextInt();
103.        adjacency_matrix = new int[number_of_vertices + 1][number_of_vertices + 1];
104.
105.        System.out.println("Enter the Weighted Matrix for the graph");
106.        for (int i = 1; i <= number_of_vertices; i++)
107.        {
108.            for (int j = 1; j <= number_of_vertices; j++)
109.            {

```

```

110.         adjacency_matrix[i][j] = scan.nextInt();
111.         if (i == j)
112.         {
113.             adjacency_matrix[i][j] = 0;
114.             continue;
115.         }
116.         if (adjacency_matrix[i][j] == 0)
117.         {
118.             adjacency_matrix[i][j] = Integer.MAX_VALUE;
119.         }
120.     }
121. }
122.
123. System.out.println("Enter the source ");
124. source = scan.nextInt();
125.
126. System.out.println("Enter the destination ");
127. destination = scan.nextInt();
128.
129. Dijkstras_Shortest_Path dijkstrasAlgorithm = new Dijkstras_Shortest_Path(
130.     number_of_vertices);
131. dijkstrasAlgorithm.dijkstra_algorithm(adjacency_matrix, source);
132.
133. System.out.println("The Shorted Path from " + source + " to " + destination + " is: ");
134. for (int i = 1; i <= dijkstrasAlgorithm.distances.length - 1; i++)
135. {
136.     if (i == destination)
137.         System.out.println(source + " to " + i + " is "
138.             + dijkstrasAlgorithm.distances[i]);
139. }
140. } catch (InputMismatchException inputMismatch)
141. {
142.     System.out.println("Wrong Input Format");
143. }
144. scan.close();
145. }
146.

```

Output:

advertisement

```
$ javac Dijkstras_Shortest_Path.java
$ java Dijkstras_Shortest_Path
```

Enter the number of vertices

5

Enter the Weighted Matrix for the graph

```
0 9 6 5 3
0 0 0 0 0
0 2 0 4 0
0 0 0 0 0
0 0 0 0 0
```

Enter the source

1

Enter the destination

4

The Shorted Path from 1 to 4 is:

1 to 4 is 5

248.Java Program to Implement Double Order Traversal of a Binary Tree

[« Prev](#)

[Next »](#)

This is a Java Program to perform Double Order traversal over binary tree. Recurse through:

1. Visit root of (sub)tree.
2. Visit left sub-tree.
3. Revisit root of (sub)tree.
4. Visit right sub-tree.

Here is the source code of the Java Program to Implement Double Order Traversal of a Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement doubleorder traversal of the Binary Search Tree
2. import java.util.Scanner;
3.
4. class BinarySearchTreeNode
5. {
6.     BinarySearchTreeNode left, right;
7.     int data;
8.
9.     public BinarySearchTreeNode()
10.    {
11.        left = null;
12.        right = null;
13.        data = 0;
14.    }
15.
16.     public BinarySearchTreeNode(int n)
17.    {
18.        left = null;
19.        right = null;
20.        data = n;
21.    }
22.
23.     public void setLeft(BinarySearchTreeNode n)
24.    {
25.        left = n;
26.    }
27.
28.     public void setRight(BinarySearchTreeNode n)
29.    {
30.        right = n;
31.    }
32.
33.     public BinarySearchTreeNode getLeft()
34.    {
35.        return left;
36.    }
37.
38.     public BinarySearchTreeNode getRight()
39.    {
40.        return right;
41.    }
42.
43.     public void setData(int d)
44.    {
45.        data = d;
46.    }
47.
```

```

48. public int getData()
49. {
50.     return data;
51. }
52.}
53.
54.class BinarySearchTree
55.{ 
56. private BinarySearchTreeNodes root;
57.
58. public BinarySearchTree()
59. {
60.     root = null;
61. }
62.
63. public boolean isEmpty()
64. {
65.     return root == null;
66. }
67.
68. public void insert(int data)
69. {
70.     root = insert(root, data);
71. }
72.
73. private BinarySearchTreeNodes insert(BinarySearchTreeNodes node, int data)
74. {
75.     if (node == null)
76.         node = new BinarySearchTreeNodes(data);
77.     else
78.     {
79.         if (data <= node.getData())
80.             node.left = insert(node.left, data);
81.         else
82.             node.right = insert(node.right, data);
83.     }
84.     return node;
85. }
86.
87. public void doubleorder()
88. {
89.     doubleorder(root);
90. }
91.
92. private void doubleorder(BinarySearchTreeNodes r)
93. {
94.     if(r != null)
95.     {
96.         System.out.print(r.getData() + " ");
97.         doubleorder(r.getLeft());
98.         System.out.print(r.getData() + " ");
99.         doubleorder(r.getRight());
100.    }
101. }
102.}
103.
104.public class Doubleorder_Traversal
105.{
106.    public static void main(String[] args)
107.    {
108.        Scanner scan = new Scanner(System.in);
109.        BinarySearchTree bst = new BinarySearchTree();
110.        System.out.println("Enter the first 10 elements of the tree\n");
111.        int N = 10;
112.        for (int i = 0; i < N; i++)
113.            bst.insert(scan.nextInt());

```

```
114.  
115.     System.out.print("\nDouble-order : ");  
116.     bst.doubleorder();  
117.  
118.     scan.close();  
119. }  
120.}
```

Output:

advertisement

```
$ javac Doubleorder_Traversal.java  
$ java Doubleorder_Traversal
```

Enter the first 10 elements of the tree

```
12 10 11 03 15 19 02 01 04 70
```

```
Double-order : 12 10 3 2 1 1 2 3 4 4 10 11 11 12 15 15 19 19 70 70
```

249.Java Program to Sort an Array of 10 Elements Using Heap Sort Algorithm

[« Prev](#)

[Next »](#)

This is a java program to perform Heap Sort. There are two types of heaps. First one is Max heap and second one is Min heap. Heap (Max/Min) is a special type of binary tree. The roots of the max heap is greater than its child roots. Other heap is Min heap it is also a special type of heap which has minimum root than his child. We can sort the array values using heap sorting algorithm. In this algorithm the heap build is used to rebuild the heap. In this example we sorting all elements of an array. The complexity of the heap sort is $O(n \cdot \log(n))$ since the method build heap takes time $O(n)$ and each of the $(n-1)$ calls to maxheap takes time $O(\lg n)$.

Here is the source code of the Java Program to Sort an Array of 10 Elements Using Heap Sort Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform sorting of randomly generated using Heap Sort
2. import java.util.Random;
3.
4. public class Heapsort
5. {
6.     private static int[] a;
7.     private static int n;
8.     private static int left;
9.     private static int right;
10.    private static int largest;
11.
12.    public static void buildheap(int[] a)
13.    {
14.        n = a.length - 1;
15.        for (int i = n / 2; i >= 0; i--)
16.        {
17.            maxheap(a, i);
18.        }
19.    }
20.
21.    public static void maxheap(int[] a, int i)
22.    {
23.        left = 2 * i;
24.        right = 2 * i + 1;
25.        if (left <= n && a[left] > a[i])
26.        {
27.            largest = left;
28.        } else
29.        {
30.            largest = i;
31.        }
32.
33.        if (right <= n && a[right] > a[largest])
34.        {
35.            largest = right;
36.        }
37.        if (largest != i)
38.        {
39.            exchange(i, largest);
40.            maxheap(a, largest);
41.        }
42.    }
43.
44.    public static void exchange(int i, int j)
45.    {
46.        int t = a[i];
47.        a[i] = a[j];
```

```

48.     a[j] = t;
49. }
50.
51. public static void sort(int[] a0)
52. {
53.     a = a0;
54.     buildheap(a);
55.
56.     for (int i = n; i > 0; i--)
57.     {
58.         exchange(0, i);
59.         n = n - 1;
60.         maxheap(a, 0);
61.     }
62. }
63.
64. public static void main(String[] args)
65. {
66.     int N = 20;
67.     int[] sequence = new int[N];
68.     Random random = new Random();
69.
70.     System.out.println("Heap Sort Test");
71.
72.     for (int i = 0; i < N; i++)
73.         sequence[i] = Math.abs(random.nextInt(100));
74.
75.     System.out.println("The original sequence is: ");
76.     for (int i = 0; i < sequence.length; i++)
77.         System.out.print(sequence[i] + " ");
78.
79.     sort(sequence);
80.
81.     System.out.println("\nThe sorted sequence is: ");
82.     for (int i = 0; i < sequence.length; i++)
83.         System.out.print(sequence[i] + " ");
84. }
85. }
```

Output:

advertisement

```
$ javac Heapsort.java
$ java Heapsort
```

```
Heap Sort Test
The original sequence is:
3 72 19 49 3 20 91 47 6 25 71 94 15 81 86 66 29 20 21 82
The sorted sequence is:
3 3 6 15 19 20 20 21 25 29 47 49 66 71 72 81 82 86 91 94
```

250.Java Program to Solve any Linear Equation in One Variable

[« Prev](#)

[Next »](#)

This is a java program to solve a linear equation in one variable.

Here is the source code of the Java Program to Solve any Linear Equation in One Variable. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the solution to the linear equation in single variable
2. import java.util.Scanner;
3.
4. public class LEquation
5. {
6.     public static void main(String args[])
7.     {
8.         String eqn = "";
9.         float ans = 0;
10.        float coeffSum = 0;
11.        float constSum = 0;
12.        float coeffx[] = new float[100];
13.        float[] constant = new float[100];
14.        Scanner in = new Scanner(System.in);
15.        System.out.println("Enter a linear equation\n");
16.        eqn = in.nextLine();
17.        eqn += "\n";
18.        // System.out.println(eqn);
19.        for (int i = 0, j = 0, k = 0; i < eqn.length() - 1;) {
20.            if (eqn.charAt(i + 1) == 'x' && i < eqn.indexOf('=')) {
21.                if (i != 0 && eqn.charAt(i - 1) == '-')
22.                {
23.                    String x = eqn.substring(i, i + 1);
24.                    if (x != "+" && x != "-")
25.                    {
26.                        int n = -(Integer.parseInt(x, 10));
27.                        coeffx[j++] = n;
28.                    }
29.                } else
30.                {
31.                    String x = eqn.substring(i, i + 1);
32.                    if (x != "+" && x != "-")
33.                    {
34.                        int n = Integer.parseInt(x, 10);
35.                        coeffx[j++] = n;
36.                    }
37.                }
38.            }
39.            i += 3;
40.        }
41.        if (eqn.charAt(i + 1) == 'x' && i > eqn.indexOf('=')) {
42.            if (eqn.charAt(i - 1) == '-')
43.            {
44.                String x = eqn.substring(i, i + 1);
45.                if (x != "+" && x != "-")
46.                {
47.                    int n = Integer.parseInt(x, 10);
48.                    coeffx[j++] = n;
49.                }
50.            } else
51.            {
52.                }
53.            }
54.        }
55.        ans = coeffSum / constSum;
56.        System.out.println("The answer is " + ans);
57.    }
58. }
```

```

54.         String x = eqn.substring(i, i + 1);
55.         if (x != "+" && x != "-")
56.         {
57.             int n = -(Integer.parseInt(x, 10));
58.             coeffx[j++] = n;
59.         }
60.     }
61.     i += 3;
62. }
63. if (eqn.charAt(i + 1) != 'x' && i < eqn.indexOf("="))
64. {
65.     if (eqn.charAt(i - 1) == '-')
66.     {
67.         String x = eqn.substring(i, i + 1);
68.         if (x != "+" && x != "-")
69.         {
70.             int n = -(Integer.parseInt(x, 10));
71.             constant[k++] = n;
72.         }
73.     } else
74.     {
75.         String x = eqn.substring(i, i + 1);
76.         if (x != "+" && x != "-")
77.         {
78.             int n = Integer.parseInt(x, 10);
79.             constant[k++] = n;
80.         }
81.     }
82.     i += 2;
83. }
84. if (eqn.charAt(i + 1) != 'x' && i > eqn.indexOf("="))
85. {
86.     if (eqn.charAt(i - 1) == '-')
87.     {
88.         String x = eqn.substring(i, i + 1);
89.         if (x != "+" && x != "-")
90.         {
91.             int n = Integer.parseInt(x, 10);
92.             constant[k++] = n;
93.         }
94.     } else
95.     {
96.         String x = eqn.substring(i, i + 1);
97.         if (x != "+" && x != "-")
98.         {
99.             int n = -(Integer.parseInt(x, 10));
100.            constant[k++] = n;
101.        }
102.    }
103.    i += 2;
104. }
105. }
106. }
107. for (int i = 0; i < coeffx.length; i++)
108.     coeffSum += coeffx[i];
109. for (int i = 0; i < constant.length; i++)
110.     constSum += constant[i];
111. ans = constSum / coeffSum;
112. System.out.println("Value of x = " + (-ans));
113. in.close();
114. }
115.
116. }

```

Output:

advertisement

```
$ javac LEquation.java  
$ java LEquation
```

Enter a linear equation

$2x+5=4x+9$

Value of x = -2.0

251.Java Program to Construct an Expression Tree for an Infix Expression

[« Prev](#)

[Next »](#)

This is a java program to construct an expression tree using infix expression and perform the infix, prefix and postfix traversal of the expression tree. The leaves of a binary expression tree are operands, such as constants or variable names, and the other nodes contain operators. These particular trees happen to be binary, because all of the operations are binary, and although this is the simplest case, it is possible for nodes to have more than two children. It is also possible for a node to have only one child, as is the case with the unary minus operator. An expression tree, T, can be evaluated by applying the operator at the root to the values obtained by recursively evaluating the left and right sub-trees.

Here is the source code of the Java Program to Construct an Expression Tree for an Infix Expression. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to construct Expression Tree using Infix Expression
2. import java.io.*;
3.
4. class Node
5. {
6.     public char data;
7.     public Node leftChild;
8.     public Node rightChild;
9.
10.    public Node(char x)
11.    {
12.        data = x;
13.    }
14.
15.    public void displayNode()
16.    {
17.        System.out.print(data);
18.    }
19.
20.
21.class Stack1
22.
23. {
24.     private Node[] a;
25.     private int top, m;
26.
27.     public Stack1(int max)
28.     {
29.         m = max;
30.         a = new Node[m];
31.         top = -1;
32.     }
33.
34.     public void push(Node key)
35.     {
36.         a[++top] = key;
37.     }
38.
39.     public Node pop()
40.     {
41.         return (a[top--]);
42.     }
43.
44.     public boolean isEmpty()
45.     {
46.         return (top == -1);
47.     }
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
```

```

46. }
47.}
48.
49.class Stack2
50.{
51. private char[] a;
52. private int top, m;
53.
54. public Stack2(int max)
55. {
56.     m = max;
57.     a = new char[m];
58.     top = -1;
59. }
60.
61. public void push(char key)
62. {
63.     a[++top] = key;
64. }
65.
66. public char pop()
67. {
68.     return (a[top--]);
69. }
70.
71. public boolean isEmpty()
72. {
73.     return (top == -1);
74. }
75.}
76.
77.class Conversion
78.{
79. private Stack2 s;
80. private String input;
81. private String output = "";
82.
83. public Conversion(String str)
84. {
85.     input = str;
86.     s = new Stack2(str.length());
87. }
88.
89. public String inToPost()
90. {
91.     for (int i = 0; i < input.length(); i++)
92.     {
93.         char ch = input.charAt(i);
94.         switch (ch)
95.         {
96.             case '+':
97.             case '-':
98.                 gotOperator(ch, 1);
99.                 break;
100.            case '*':
101.            case '/':
102.                 gotOperator(ch, 2);
103.                 break;
104.            case '(':
105.                 s.push(ch);
106.                 break;
107.            case ')':
108.                 gotParenthesis();
109.                 break;
110.            default:
111.                 output = output + ch;

```

```

112.     }
113.   }
114.   while (!s.isEmpty())
115.     output = output + s.pop();
116.   return output;
117. }
118.
119. private void gotOperator(char opThis, int prec1)
120. {
121.   while (!s.isEmpty())
122.   {
123.     char opTop = s.pop();
124.     if (opTop == '(')
125.     {
126.       s.push(opTop);
127.       break;
128.     } else
129.     {
130.       int prec2;
131.       if (opTop == '+' || opTop == '-')
132.         prec2 = 1;
133.       else
134.         prec2 = 2;
135.       if (prec2 < prec1)
136.       {
137.         s.push(opTop);
138.         break;
139.       } else
140.         output = output + opTop;
141.     }
142.   }
143.   s.push(opThis);
144. }
145.
146. private void gotParenthesis()
147. {
148.   while (!s.isEmpty())
149.   {
150.     char ch = s.pop();
151.     if (ch == '(')
152.       break;
153.     else
154.       output = output + ch;
155.   }
156. }
157. }
158.
159.class Tree
160.
161. private Node root;
162.
163. public Tree()
164. {
165.   root = null;
166. }
167.
168. public void insert(String s)
169. {
170.   Conversion c = new Conversion(s);
171.   s = c.inToPost();
172.   Stack1 stk = new Stack1(s.length());
173.   s = s + "#";
174.   int i = 0;
175.   char symbol = s.charAt(i);
176.   Node newNode;
177.   while (symbol != '#')

```

```

178.     {
179.         if (symbol >= '0' && symbol <= '9' || symbol >= 'A'
180.             && symbol <= 'Z' || symbol >= 'a' && symbol <= 'z')
181.         {
182.             newNode = new Node(symbol);
183.             stk.push(newNode);
184.         } else if (symbol == '+' || symbol == '-' || symbol == '/'
185.             || symbol == '*')
186.         {
187.             Node ptr1 = stk.pop();
188.             Node ptr2 = stk.pop();
189.             newNode = new Node(symbol);
190.             newNode.leftChild = ptr2;
191.             newNode.rightChild = ptr1;
192.             stk.push(newNode);
193.         }
194.         symbol = s.charAt(++i);
195.     }
196.     root = stk.pop();
197. }
198.
199. public void traverse(int type)
200. {
201.     switch (type)
202.     {
203.         case 1:
204.             System.out.print("Preorder Traversal:- ");
205.             preOrder(root);
206.             break;
207.         case 2:
208.             System.out.print("Inorder Traversal:- ");
209.             inOrder(root);
210.             break;
211.         case 3:
212.             System.out.print("Postorder Traversal:- ");
213.             postOrder(root);
214.             break;
215.         default:
216.             System.out.println("Invalid Choice");
217.     }
218. }
219.
220. private void preOrder(Node localRoot)
221. {
222.     if (localRoot != null)
223.     {
224.         localRoot.displayNode();
225.         preOrder(localRoot.leftChild);
226.         preOrder(localRoot.rightChild);
227.     }
228. }
229.
230. private void inOrder(Node localRoot)
231. {
232.     if (localRoot != null)
233.     {
234.         inOrder(localRoot.leftChild);
235.         localRoot.displayNode();
236.         inOrder(localRoot.rightChild);
237.     }
238. }
239.
240. private void postOrder(Node localRoot)
241. {
242.     if (localRoot != null)
243.     {

```

```

244.     postOrder(localRoot.leftChild);
245.     postOrder(localRoot.rightChild);
246.     localRoot.displayNode();
247.   }
248. }
249. }
250.
251.public class Infix_Expression_Tree
252. {
253.   public static void main(String args[]) throws IOException
254.   {
255.     String ch = "y";
256.     DataInputStream inp = new DataInputStream(System.in);
257.     while (ch.equals("y"))
258.     {
259.       Tree t1 = new Tree();
260.       System.out.println("Enter the Expression");
261.       String a = inp.readLine();
262.       t1.insert(a);
263.       t1.traverse(1);
264.       System.out.println("");
265.       t1.traverse(2);
266.       System.out.println("");
267.       t1.traverse(3);
268.       System.out.println("");
269.       System.out.print("Enter y to continue ");
270.       ch = inp.readLine();
271.     }
272.   }
273. }
```

Output:

advertisement

```
$ javac Infix_Expression_Tree.java
$ java Infix_Expression_Tree
```

Enter the Expression
A+B*C-D

Preorder Traversal:- -+A*BCD
 Inorder Traversal:- A+B*C-D
 Postorder Traversal:- ABC*+D-

252.Java Program to Construct an Expression Tree for an Postfix Expression

[« Prev](#)

[Next »](#)

This is a java program to construct an expression tree using postfix expression and perform the infix, prefix and postfix traversal of the expression tree. The leaves of a binary expression tree are operands, such as constants or variable names, and the other nodes contain operators. These particular trees happen to be binary, because all of the operations are binary, and although this is the simplest case, it is possible for nodes to have more than two children. It is also possible for a node to have only one child, as is the case with the unary minus operator. An expression tree, T, can be evaluated by applying the operator at the root to the values obtained by recursively evaluating the left and right subtrees.

Here is the source code of the Java Program to Construct an Expression Tree for an Postfix Expression. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to construct Expression Tree using Postfix Expression
2. import java.io.*;
3.
4. class Node
5. {
6.     public char data;
7.     public Node leftChild;
8.     public Node rightChild;
9.
10.    public Node(char x)
11.    {
12.        data = x;
13.    }
14.
15.    public void displayNode()
16.    {
17.        System.out.print(data);
18.    }
19.
20.
21.class Stack1
22.
23. {
24.     private Node[] a;
25.     private int top, m;
26.
27.     public Stack1(int max)
28.     {
29.         m = max;
30.         a = new Node[m];
31.         top = -1;
32.     }
33.
34.     public void push(Node key)
35.     {
36.         a[++top] = key;
37.     }
38.
39.     public Node pop()
40.     {
41.         return (a[top--]);
42.     }
43.
44.     public boolean isEmpty()
45.     {
46.         return (top == -1);
47.     }
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
```

```

46. }
47.}
48.
49.class Stack2
50.{
51. private char[] a;
52. private int top, m;
53.
54. public Stack2(int max)
55. {
56.     m = max;
57.     a = new char[m];
58.     top = -1;
59. }
60.
61. public void push(char key)
62. {
63.     a[++top] = key;
64. }
65.
66. public char pop()
67. {
68.     return (a[top--]);
69. }
70.
71. public boolean isEmpty()
72. {
73.     return (top == -1);
74. }
75.}
76.
77.class Conversion
78.{
79. private Stack2 s;
80. private String input;
81. private String output = "";
82.
83. public Conversion(String str)
84. {
85.     input = str;
86.     s = new Stack2(str.length());
87. }
88.
89. public String inToPost()
90. {
91.     for (int i = 0; i < input.length(); i++)
92.     {
93.         char ch = input.charAt(i);
94.         switch (ch)
95.         {
96.             case '+':
97.             case '-':
98.                 gotOperator(ch, 1);
99.                 break;
100.            case '*':
101.            case '/':
102.                 gotOperator(ch, 2);
103.                 break;
104.            case '(':
105.                 s.push(ch);
106.                 break;
107.            case ')':
108.                 gotParenthesis();
109.                 break;
110.            default:
111.                 output = output + ch;

```

```

112.     }
113.   }
114.   while (!s.isEmpty())
115.     output = output + s.pop();
116.   return output;
117. }
118.
119. private void gotOperator(char opThis, int prec1)
120. {
121.   while (!s.isEmpty())
122.   {
123.     char opTop = s.pop();
124.     if (opTop == '(')
125.     {
126.       s.push(opTop);
127.       break;
128.     } else
129.     {
130.       int prec2;
131.       if (opTop == '+' || opTop == '-')
132.         prec2 = 1;
133.       else
134.         prec2 = 2;
135.       if (prec2 < prec1)
136.       {
137.         s.push(opTop);
138.         break;
139.       } else
140.         output = output + opTop;
141.     }
142.   }
143.   s.push(opThis);
144. }
145.
146. private void gotParenthesis()
147. {
148.   while (!s.isEmpty())
149.   {
150.     char ch = s.pop();
151.     if (ch == '(')
152.       break;
153.     else
154.       output = output + ch;
155.   }
156. }
157. }
158.
159.class Tree
160.
161. private Node root;
162.
163. public Tree()
164. {
165.   root = null;
166. }
167.
168. public void insert(String s)
169. {
170.   Conversion c = new Conversion(s);
171.   s = c.inToPost();
172.   Stack1 stk = new Stack1(s.length());
173.   s = s + "#";
174.   int i = 0;
175.   char symbol = s.charAt(i);
176.   Node newNode;
177.   while (symbol != '#')

```

```

178.     {
179.         if (symbol >= '0' && symbol <= '9' || symbol >= 'A'
180.             && symbol <= 'Z' || symbol >= 'a' && symbol <= 'z')
181.         {
182.             newNode = new Node(symbol);
183.             stk.push(newNode);
184.         } else if (symbol == '+' || symbol == '-' || symbol == '/'
185.             || symbol == '*')
186.         {
187.             Node ptr1 = stk.pop();
188.             Node ptr2 = stk.pop();
189.             newNode = new Node(symbol);
190.             newNode.leftChild = ptr2;
191.             newNode.rightChild = ptr1;
192.             stk.push(newNode);
193.         }
194.         symbol = s.charAt(++i);
195.     }
196.     root = stk.pop();
197. }
198.
199. public void traverse(int type)
200. {
201.     switch (type)
202.     {
203.         case 1:
204.             System.out.print("Preorder Traversal:- ");
205.             preOrder(root);
206.             break;
207.         case 2:
208.             System.out.print("Inorder Traversal:- ");
209.             inOrder(root);
210.             break;
211.         case 3:
212.             System.out.print("Postorder Traversal:- ");
213.             postOrder(root);
214.             break;
215.         default:
216.             System.out.println("Invalid Choice");
217.     }
218. }
219.
220. private void preOrder(Node localRoot)
221. {
222.     if (localRoot != null)
223.     {
224.         localRoot.displayNode();
225.         preOrder(localRoot.leftChild);
226.         preOrder(localRoot.rightChild);
227.     }
228. }
229.
230. private void inOrder(Node localRoot)
231. {
232.     if (localRoot != null)
233.     {
234.         inOrder(localRoot.leftChild);
235.         localRoot.displayNode();
236.         inOrder(localRoot.rightChild);
237.     }
238. }
239.
240. private void postOrder(Node localRoot)
241. {
242.     if (localRoot != null)
243.     {

```

```

244.     postOrder(localRoot.leftChild);
245.     postOrder(localRoot.rightChild);
246.     localRoot.displayNode();
247.   }
248. }
249. }
250.
251.public class Postfix_Expression_Tree
252. {
253.   public static void main(String args[]) throws IOException
254.   {
255.     String ch = "y";
256.     DataInputStream inp = new DataInputStream(System.in);
257.     while (ch.equals("y"))
258.     {
259.       Tree t1 = new Tree();
260.       System.out.println("Enter the Expression");
261.       String a = inp.readLine();
262.       t1.insert(a);
263.       t1.traverse(1);
264.       System.out.println("");
265.       t1.traverse(2);
266.       System.out.println("");
267.       t1.traverse(3);
268.       System.out.println("");
269.       System.out.print("Enter y to continue ");
270.       ch = inp.readLine();
271.     }
272.   }
273. }
```

Output:

advertisement

```
$ javac Postfix_Expression_Tree.java
$ java Postfix_Expression_Tree
```

Enter the Expression
ABC*D-

Preorder Traversal:- -+A*BCD
 Inorder Traversal:- A+B*C-D
 Postorder Traversal:- ABC*D-

253.Java Program to Construct an Expression Tree for an Prefix Expression

[« Prev](#)

[Next »](#)

This is a java program to construct an expression tree using prefix expression and perform the infix, prefix and postfix traversal of the expression tree. The leaves of a binary expression tree are operands, such as constants or variable names, and the other nodes contain operators. These particular trees happen to be binary, because all of the operations are binary, and although this is the simplest case, it is possible for nodes to have more than two children. It is also possible for a node to have only one child, as is the case with the unary minus operator. An expression tree, T, can be evaluated by applying the operator at the root to the values obtained by recursively evaluating the left and right subtrees.

Here is the source code of the Java Program to Construct an Expression Tree for an Prefix Expression. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to construct Expression Tree using Prefix Expression
2. import java.io.*;
3.
4. class Node
5. {
6.     public char data;
7.     public Node leftChild;
8.     public Node rightChild;
9.
10.    public Node(char x)
11.    {
12.        data = x;
13.    }
14.
15.    public void displayNode()
16.    {
17.        System.out.print(data);
18.    }
19.
20.
21.class Stack1
22.
23. {
24.     private Node[] a;
25.     private int top, m;
26.
27.     public Stack1(int max)
28.     {
29.         m = max;
30.         a = new Node[m];
31.         top = -1;
32.     }
33.
34.     public void push(Node key)
35.     {
36.         a[++top] = key;
37.     }
38.
39.     public Node pop()
40.     {
41.         return (a[top--]);
42.     }
43.
44.     public boolean isEmpty()
45.     {
46.         return (top == -1);
```

```

46. }
47.}
48.
49.class Stack2
50.{
51. private char[] a;
52. private int top, m;
53.
54. public Stack2(int max)
55. {
56.     m = max;
57.     a = new char[m];
58.     top = -1;
59. }
60.
61. public void push(char key)
62. {
63.     a[++top] = key;
64. }
65.
66. public char pop()
67. {
68.     return (a[top--]);
69. }
70.
71. public boolean isEmpty()
72. {
73.     return (top == -1);
74. }
75.}
76.
77.class Conversion
78.{
79. private Stack2 s;
80. private String input;
81. private String output = "";
82.
83. public Conversion(String str)
84. {
85.     input = str;
86.     s = new Stack2(str.length());
87. }
88.
89. public String inToPost()
90. {
91.     for (int i = 0; i < input.length(); i++)
92.     {
93.         char ch = input.charAt(i);
94.         switch (ch)
95.         {
96.             case '+':
97.             case '-':
98.                 gotOperator(ch, 1);
99.                 break;
100.            case '*':
101.            case '/':
102.                 gotOperator(ch, 2);
103.                 break;
104.            case '(':
105.                 s.push(ch);
106.                 break;
107.            case ')':
108.                 gotParenthesis();
109.                 break;
110.            default:
111.                 output = output + ch;

```

```

112.     }
113.   }
114.   while (!s.isEmpty())
115.     output = output + s.pop();
116.   return output;
117. }
118.
119. private void gotOperator(char opThis, int prec1)
120. {
121.   while (!s.isEmpty())
122.   {
123.     char opTop = s.pop();
124.     if (opTop == '(')
125.     {
126.       s.push(opTop);
127.       break;
128.     } else
129.     {
130.       int prec2;
131.       if (opTop == '+' || opTop == '-')
132.         prec2 = 1;
133.       else
134.         prec2 = 2;
135.       if (prec2 < prec1)
136.       {
137.         s.push(opTop);
138.         break;
139.       } else
140.         output = output + opTop;
141.     }
142.   }
143.   s.push(opThis);
144. }
145.
146. private void gotParenthesis()
147. {
148.   while (!s.isEmpty())
149.   {
150.     char ch = s.pop();
151.     if (ch == '(')
152.       break;
153.     else
154.       output = output + ch;
155.   }
156. }
157. }
158.
159.class Tree
160.
161. private Node root;
162.
163. public Tree()
164. {
165.   root = null;
166. }
167.
168. public void insert(String s)
169. {
170.   Conversion c = new Conversion(s);
171.   s = c.inToPost();
172.   Stack1 stk = new Stack1(s.length());
173.   s = s + "#";
174.   int i = 0;
175.   char symbol = s.charAt(i);
176.   Node newNode;
177.   while (symbol != '#')

```

```

178.     {
179.         if (symbol >= '0' && symbol <= '9' || symbol >= 'A'
180.             && symbol <= 'Z' || symbol >= 'a' && symbol <= 'z')
181.         {
182.             newNode = new Node(symbol);
183.             stk.push(newNode);
184.         } else if (symbol == '+' || symbol == '-' || symbol == '/'
185.             || symbol == '*')
186.         {
187.             Node ptr1 = stk.pop();
188.             Node ptr2 = stk.pop();
189.             newNode = new Node(symbol);
190.             newNode.leftChild = ptr2;
191.             newNode.rightChild = ptr1;
192.             stk.push(newNode);
193.         }
194.         symbol = s.charAt(++i);
195.     }
196.     root = stk.pop();
197. }
198.
199. public void traverse(int type)
200. {
201.     switch (type)
202.     {
203.         case 1:
204.             System.out.print("Preorder Traversal:- ");
205.             preOrder(root);
206.             break;
207.         case 2:
208.             System.out.print("Inorder Traversal:- ");
209.             inOrder(root);
210.             break;
211.         case 3:
212.             System.out.print("Postorder Traversal:- ");
213.             postOrder(root);
214.             break;
215.         default:
216.             System.out.println("Invalid Choice");
217.     }
218. }
219.
220. private void preOrder(Node localRoot)
221. {
222.     if (localRoot != null)
223.     {
224.         localRoot.displayNode();
225.         preOrder(localRoot.leftChild);
226.         preOrder(localRoot.rightChild);
227.     }
228. }
229.
230. private void inOrder(Node localRoot)
231. {
232.     if (localRoot != null)
233.     {
234.         inOrder(localRoot.leftChild);
235.         localRoot.displayNode();
236.         inOrder(localRoot.rightChild);
237.     }
238. }
239.
240. private void postOrder(Node localRoot)
241. {
242.     if (localRoot != null)
243.     {

```

```

244.     postOrder(localRoot.leftChild);
245.     postOrder(localRoot.rightChild);
246.     localRoot.displayNode();
247.   }
248. }
249. }
250.
251.public class Prefix_Expression_Tree
252. {
253.   public static void main(String args[]) throws IOException
254.   {
255.     String ch = "y";
256.     DataInputStream inp = new DataInputStream(System.in);
257.     while (ch.equals("y"))
258.     {
259.       Tree t1 = new Tree();
260.       System.out.println("Enter the Expression");
261.       String a = inp.readLine();
262.       t1.insert(a);
263.       t1.traverse(1);
264.       System.out.println("");
265.       t1.traverse(2);
266.       System.out.println("");
267.       t1.traverse(3);
268.       System.out.println("");
269.       System.out.print("Enter y to continue ");
270.       ch = inp.readLine();
271.     }
272.   }
273. }
```

Output:

advertisement

```
$ javac Prefix_Expression_Tree.java
$ java Prefix_Expression_Tree
```

Enter the Expression
-+A*BCD

Preorder Traversal:- -+A*BCD
 Inorder Traversal:- A+B*C-D
 Postorder Traversal:- ABC*+D-

254. Java Program to Perform Inorder Non-Recursive Traversal of a Given Binary Tree

« Prev

[Next »](#)

This is a java program to construct a binary tree and perform in-order traversal of the constructed binary tree.
Nodes visited are in the order:
visit Left node
visit Root node
visit Right node

Here is the source code of the Java Program to Perform Inorder Non-Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement non recursive in order traversal of Binary Search Tree
2. import java.util.Scanner;
3. import java.util.Stack;
4.
5. class BinarySearchTreeNode
6. {
7.     BinarySearchTreeNode left, right;
8.     int data;
9.
10.    public BinarySearchTreeNode()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BinarySearchTreeNode(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BinarySearchTreeNode n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BinarySearchTreeNode n)
30.    {
31.        right = n;
32.    }
33.
34.    public BinarySearchTreeNode getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BinarySearchTreeNode getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
48.
```

```
48.
49. public int getData()
50. {
51.     return data;
52. }
53.
54.
55.class BinarySearchTreeOperations
56.
57. private BinarySearchTreeNode root;
58.
59. public BinarySearchTreeOperations()
60. {
61.     root = null;
62. }
63.
64. public boolean isEmpty()
65. {
66.     return root == null;
67. }
68.
69. public void insert(int data)
70. {
71.     root = insert(root, data);
72. }
73.
74. private BinarySearchTreeNode insert(BinarySearchTreeNode node, int data)
75. {
76.     if (node == null)
77.         node = new BinarySearchTreeNode(data);
78.     else
79.     {
80.         if (data <= node.getData())
81.             node.left = insert(node.left, data);
82.         else
83.             node.right = insert(node.right, data);
84.     }
85.     return node;
86. }
87.
88. public void inorder()
89. {
90.     inorder(root);
91. }
92.
93. private void inorder(BinarySearchTreeNode r)
94. {
95.     if (r == null)
96.         return;
97.
98.     Stack<BinarySearchTreeNode> stack = new Stack<BinarySearchTreeNode>();
99.
100.    while (!stack.isEmpty() || r != null)
101.    {
102.        if (r != null)
103.        {
104.            stack.push(r);
105.            r = r.left;
106.        } else
107.        {
108.            r = stack.pop();
109.            System.out.print(r.data + " ");
110.            r = r.right;
111.        }
112.    }
113. }
```

```
114. }
115.
116.public class Inorder_NonRecursive_BST
117. {
118.   public static void main(String[] args)
119.   {
120.     Scanner scan = new Scanner(System.in);
121.     BinarySearchTreeOperations bst = new BinarySearchTreeOperations();
122.     System.out.println("Enter the first 10 elements of the tree\n");
123.     int N = 10;
124.     for (int i = 0; i < N; i++)
125.       bst.insert(scan.nextInt());
126.
127.     System.out.print("\nIn order : ");
128.     bst.inorder();
129.
130.     scan.close();
131.   }
132. }
```

Output:

advertisement

```
$ javac Inorder_NonRecursive_BST.java
$ java Inorder_NonRecursive_BST
```

Enter the first 10 elements of the tree

12 4 10 13 15 46 78 98 45 12

In order : 4 10 12 12 13 15 45 46 78 98

255. Java Program to Perform Preorder Non-Recursive Traversal of a Given Binary Tree

[« Prev](#)

[Next »](#)

Here is the source code of the Java Program to Perform Preorder Non-Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement non recursive pre-order traversal of Binary Search Tree
2. import java.util.Scanner;
3. import java.util.Stack;
4.
5. class BinarySearchTreeNode
6. {
7.     BinarySearchTreeNode left, right;
8.     int data;
9.
10.    public BinarySearchTreeNode()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BinarySearchTreeNode(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BinarySearchTreeNode n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BinarySearchTreeNode n)
30.    {
31.        right = n;
32.    }
33.
34.    public BinarySearchTreeNode getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BinarySearchTreeNode getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
```

```

48.
49. public int getData()
50. {
51.     return data;
52. }
53.
54.
55.class BinarySearchTreeOperations
56.{
57.    private BinarySearchTreeNode root;
58.
59.    public BinarySearchTreeOperations()
60.    {
61.        root = null;
62.    }
63.
64.    public boolean isEmpty()
65.    {
66.        return root == null;
67.    }
68.
69.    public void insert(int data)
70.    {
71.        root = insert(root, data);
72.    }
73.
74.    private BinarySearchTreeNode insert(BinarySearchTreeNode node, int data)
75.    {
76.        if (node == null)
77.            node = new BinarySearchTreeNode(data);
78.        else
79.        {
80.            if (data <= node.getData())
81.                node.left = insert(node.left, data);
82.            else
83.                node.right = insert(node.right, data);
84.        }
85.        return node;
86.    }
87.
88.    public void preorder()
89.    {
90.        preorder(root);
91.    }
92.
93.    private void preorder(BinarySearchTreeNode r)
94.    {
95.        Stack<BinarySearchTreeNode> s = new Stack<BinarySearchTreeNode>();
96.        s.push(r);
97.        while (!s.empty())
98.        {
99.            r = s.pop();
100.            System.out.print(r.data + " ");
101.            if (r.right != null)
102.                s.push(r.right);
103.            if (r.left != null)
104.                s.push(r.left);
105.        }
106.    }
107.}
108.
109.public class Preorder_NonRecursive_BST
110.{ 
111.    public static void main(String[] args)
112.    {
113.        Scanner scan = new Scanner(System.in);

```

```
114. BinarySearchTreeOperations bst = new BinarySearchTreeOperations();
115. System.out.println("Enter the first 10 elements of the tree\n");
116. int N = 10;
117. for (int i = 0; i < N; i++)
118.     bst.insert(scan.nextInt());
119.
120. System.out.print("\nPre order : ");
121. bst.preorder();
122.
123. scan.close();
124. }
125. }
```

Output:

advertisement

```
$ javac Preorder_NonRecursive_BST.java
$ java Preorder_NonRecursive_BST
```

Enter the first 10 elements of the tree

12 4 10 13 15 46 78 98 45 12

Pre order : 12 4 10 12 13 15 46 45 78 98

256. Java Program to Perform Postorder Non-Recursive Traversal of a Given Binary Tree

« Prev

[Next »](#)

This is a java program to construct a binary tree and perform postorder traversal of the constructed binary tree.
Nodes visited are in the order:
visit Left node
visit Right node
visit Root node

Here is the source code of the Java Program to Perform Postorder Non-Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement non recursive post order traversal of Binary Search Tree
2. import java.util.Scanner;
3. import java.util.Stack;
4.
5. class BinarySearchTreeNode
6. {
7.     BinarySearchTreeNode left, right;
8.     int data;
9.
10.    public BinarySearchTreeNode()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BinarySearchTreeNode(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BinarySearchTreeNode n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BinarySearchTreeNode n)
30.    {
31.        right = n;
32.    }
33.
34.    public BinarySearchTreeNode getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BinarySearchTreeNode getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
48.
```

```

48.
49. public int getData()
50. {
51.     return data;
52. }
53.
54.
55.class BinarySearchTreeOperations
56.{
57.    private BinarySearchTreeNode root;
58.
59.    public BinarySearchTreeOperations()
60.    {
61.        root = null;
62.    }
63.
64.    public boolean isEmpty()
65.    {
66.        return root == null;
67.    }
68.
69.    public void insert(int data)
70.    {
71.        root = insert(root, data);
72.    }
73.
74.    private BinarySearchTreeNode insert(BinarySearchTreeNode node, int data)
75.    {
76.        if (node == null)
77.            node = new BinarySearchTreeNode(data);
78.        else
79.        {
80.            if (data <= node.getData())
81.                node.left = insert(node.left, data);
82.            else
83.                node.right = insert(node.right, data);
84.        }
85.        return node;
86.    }
87.
88.    public void postorder()
89.    {
90.        postorder(root);
91.    }
92.
93.    private void postorder(BinarySearchTreeNode r)
94.    {
95.        if (root == null)
96.            return;
97.
98.        final Stack<BinarySearchTreeNode> stack = new Stack<BinarySearchTreeNode>();
99.        BinarySearchTreeNode node = root;
100.
101.       while (!stack.isEmpty() || node != null)
102.       {
103.           while (node != null)
104.           {
105.               if (node.right != null)
106.                   stack.add(node.right);
107.                   stack.add(node);
108.                   node = node.left;
109.               }
110.
111.           node = stack.pop();
112.
113.           if (node.right != null && !stack.isEmpty())

```

```

114.         && node.right == stack.peek())
115.     {
116.         BinarySearchTreeNodes nodeRight = stack.pop();
117.         stack.push(node);
118.         node = nodeRight;
119.     } else
120.     {
121.         System.out.print(node.data + " ");
122.         node = null;
123.     }
124. }
125. }
126. }
127.
128.public class Postorder_NonRecursive_BST
129. {
130.     public static void main(String[] args)
131.     {
132.         Scanner scan = new Scanner(System.in);
133.         BinarySearchTreeOperations bst = new BinarySearchTreeOperations();
134.         System.out.println("Enter the first 10 elements of the tree\n");
135.         int N = 10;
136.         for (int i = 0; i < N; i++)
137.             bst.insert(scan.nextInt());
138.
139.         System.out.print("\nPost order : ");
140.         bst.postorder();
141.
142.         scan.close();
143.     }
144. }
```

Output:

advertisement

```
$ javac Postorder_NonRecursive_BST.java
$ java Postorder_NonRecursive_BST
```

Enter the first 10 elements of the tree

12 4 10 13 15 46 78 98 45 12

Post order : 12 10 4 45 98 78 46 15 13 12

257. Java Program to Perform Inorder Recursive Traversal of a Given Binary Tree

« Prev

[Next »](#)

Here is the source code of the Java Program to Perform Inorder Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement recursive inorder traversal of the Binary Search Tree
2. import java.util.Scanner;
3.
4. class BinarySearchTreeNode
5. {
6.     BinarySearchTreeNode left, right;
7.     int data;
8.
9.     public BinarySearchTreeNode()
10.    {
11.        left = null;
12.        right = null;
13.        data = 0;
14.    }
15.
16.    public BinarySearchTreeNode(int n)
17.    {
18.        left = null;
19.        right = null;
20.        data = n;
21.    }
22.
23.    public void setLeft(BinarySearchTreeNode n)
24.    {
25.        left = n;
26.    }
27.
28.    public void setRight(BinarySearchTreeNode n)
29.    {
30.        right = n;
31.    }
32.
33.    public BinarySearchTreeNode getLeft()
34.    {
35.        return left;
36.    }
37.
38.    public BinarySearchTreeNode getRight()
39.    {
40.        return right;
41.    }
42.
43.    public void setData(int d)
44.    {
45.        data = d;
46.    }
47.
48.    public int getData()
```

```

49.  {
50.      return data;
51.  }
52.}
53.
54.class BinarySearchTree
55.{
56.    private BinarySearchTreeNode root;
57.
58.    public BinarySearchTree()
59.    {
60.        root = null;
61.    }
62.
63.    public boolean isEmpty()
64.    {
65.        return root == null;
66.    }
67.
68.    public void insert(int data)
69.    {
70.        root = insert(root, data);
71.    }
72.
73.    private BinarySearchTreeNode insert(BinarySearchTreeNode node, int data)
74.    {
75.        if (node == null)
76.            node = new BinarySearchTreeNode(data);
77.        else
78.        {
79.            if (data <= node.getData())
80.                node.left = insert(node.left, data);
81.            else
82.                node.right = insert(node.right, data);
83.        }
84.        return node;
85.    }
86.
87.    public void inorder()
88.    {
89.        inorder(root);
90.    }
91.
92.    private void inorder(BinarySearchTreeNode r)
93.    {
94.        if (r != null)
95.        {
96.            inorder(r.getLeft());
97.            System.out.print(r.getData() + " ");
98.            inorder(r.getRight());
99.        }
100.    }
101.}
102.
103.public class Inorder_Recursive_BST
104.{
105.    public static void main(String[] args)
106.    {
107.        Scanner scan = new Scanner(System.in);
108.        BinarySearchTree bst = new BinarySearchTree();
109.        System.out.println("Enter the first 10 elements of the tree\n");
110.        int N = 10;
111.        for (int i = 0; i < N; i++)
112.            bst.insert(scan.nextInt());
113.
114.        System.out.print("\nIn order : ");

```

```
115.     bst.inorder();
116.
117.     scan.close();
118. }
119. }
```

Output:

advertisement

```
$ javac Inorder_Recursive_BST.java
$ java Inorder_Recursive_BST
```

Enter the first 10 elements of the tree

```
12 10 11 03 15 19 02 01 04 70
```

```
In order : 1 2 3 4 10 11 12 15 19 70
```

258. Java Program to Perform Postorder Recursive Traversal of a Given Binary Tree

« Prev

[Next »](#)

This is a java program to construct a binary tree and perform postorder traversal of the constructed binary tree.
Nodes visited are in the order:
visit Left node
visit Right node
visit Root node

Here is the source code of the Java Program to Perform Postorder Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement recursive postorder traversal of the Binary Search Tree
2. import java.util.Scanner;
3.
4. class BinarySearchTreeNode
5. {
6.     BinarySearchTreeNode left, right;
7.     int data;
8.
9.     public BinarySearchTreeNode()
10.    {
11.        left = null;
12.        right = null;
13.        data = 0;
14.    }
15.
16.    public BinarySearchTreeNode(int n)
17.    {
18.        left = null;
19.        right = null;
20.        data = n;
21.    }
22.
23.    public void setLeft(BinarySearchTreeNode n)
24.    {
25.        left = n;
26.    }
27.
28.    public void setRight(BinarySearchTreeNode n)
29.    {
30.        right = n;
31.    }
32.
33.    public BinarySearchTreeNode getLeft()
34.    {
35.        return left;
36.    }
37.
38.    public BinarySearchTreeNode getRight()
39.    {
40.        return right;
41.    }
42.
43.    public void setData(int d)
44.    {
45.        data = d;
46.    }
47.
48.    public int getData()
```

```

49.  {
50.      return data;
51.  }
52.}
53.
54.class BinarySearchTree
55.{
56.    private BinarySearchTreeNodes root;
57.
58.    public BinarySearchTree()
59.    {
60.        root = null;
61.    }
62.
63.    public boolean isEmpty()
64.    {
65.        return root == null;
66.    }
67.
68.    public void insert(int data)
69.    {
70.        root = insert(root, data);
71.    }
72.
73.    private BinarySearchTreeNodes insert(BinarySearchTreeNodes node, int data)
74.    {
75.        if (node == null)
76.            node = new BinarySearchTreeNodes(data);
77.        else
78.        {
79.            if (data <= node.getData())
80.                node.left = insert(node.left, data);
81.            else
82.                node.right = insert(node.right, data);
83.        }
84.        return node;
85.    }
86.
87.    public void postorder()
88.    {
89.        postorder(root);
90.    }
91.
92.    private void postorder(BinarySearchTreeNodes r)
93.    {
94.        if (r != null)
95.        {
96.            postorder(r.getLeft());
97.            postorder(r.getRight());
98.            System.out.print(r.getData() + " ");
99.        }
100.    }
101.}
102.
103.public class Postorder_Recursive_BST
104.{
105.    public static void main(String[] args)
106.    {
107.        Scanner scan = new Scanner(System.in);
108.        BinarySearchTree bst = new BinarySearchTree();
109.        System.out.println("Enter the first 10 elements of the tree\n");
110.        int N = 10;
111.        for (int i = 0; i < N; i++)
112.            bst.insert(scan.nextInt());
113.
114.        System.out.print("\nPost order : ");

```

```
115.     bst.postorder();
116.
117.     scan.close();
118. }
119. }
```

Output:

advertisement

```
$ javac Postorder_Recursive_BST.java
$ java Postorder_Recursive_BST
```

Enter the first 10 elements of the tree

```
12 10 11 03 15 19 02 01 04 70
```

```
Post order : 1 2 4 3 11 10 70 19 15 12
```

259. Java Program to Perform Preorder Recursive Traversal of a Given Binary Tree

[« Prev](#)

[Next »](#)

Here is the source code of the Java Program to Perform Preorder Recursive Traversal of a Given Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement recursive preorder traversal of the Binary Search Tree
2. import java.util.Scanner;
3.
4. class BinarySearchTreeNode
5. {
6.     BinarySearchTreeNode left, right;
7.     int data;
8.
9.     public BinarySearchTreeNode()
10.    {
11.        left = null;
12.        right = null;
13.        data = 0;
14.    }
15.
16.    public BinarySearchTreeNode(int n)
17.    {
18.        left = null;
19.        right = null;
20.        data = n;
21.    }
22.
23.    public void setLeft(BinarySearchTreeNode n)
24.    {
25.        left = n;
26.    }
27.
28.    public void setRight(BinarySearchTreeNode n)
29.    {
30.        right = n;
31.    }
32.
33.    public BinarySearchTreeNode getLeft()
34.    {
35.        return left;
36.    }
37.
38.    public BinarySearchTreeNode getRight()
39.    {
40.        return right;
41.    }
42.
43.    public void setData(int d)
44.    {
45.        data = d;
46.    }
47.
48.    public int getData()
```

```

49.  {
50.      return data;
51.  }
52.}
53.
54.class BinarySearchTree
55.{
56.    private BinarySearchTreeNode root;
57.
58.    public BinarySearchTree()
59.    {
60.        root = null;
61.    }
62.
63.    public boolean isEmpty()
64.    {
65.        return root == null;
66.    }
67.
68.    public void insert(int data)
69.    {
70.        root = insert(root, data);
71.    }
72.
73.    private BinarySearchTreeNode insert(BinarySearchTreeNode node, int data)
74.    {
75.        if (node == null)
76.            node = new BinarySearchTreeNode(data);
77.        else
78.        {
79.            if (data <= node.getData())
80.                node.left = insert(node.left, data);
81.            else
82.                node.right = insert(node.right, data);
83.        }
84.        return node;
85.    }
86.
87.    public void preorder()
88.    {
89.        preorder(root);
90.    }
91.
92.    private void preorder(BinarySearchTreeNode r)
93.    {
94.        if (r != null)
95.        {
96.            System.out.print(r.getData() + " ");
97.            preorder(r.getLeft());
98.            preorder(r.getRight());
99.        }
100.    }
101.}
102.
103.public class Preorder_Recursive_BST
104.{
105.    public static void main(String[] args)
106.    {
107.        Scanner scan = new Scanner(System.in);
108.        BinarySearchTree bst = new BinarySearchTree();
109.        System.out.println("Enter the first 10 elements of the tree\n");
110.        int N = 10;
111.        for (int i = 0; i < N; i++)
112.            bst.insert(scan.nextInt());
113.
114.        System.out.print("\nPre order : ");

```

```
115.     bst.preorder();
116.
117.     scan.close();
118. }
119. }
```

Output:

advertisement

```
$ javac Preorder_Recursive_BST.java
$ java Preorder_Recursive_BST
```

Enter the first 10 elements of the tree

```
12 10 11 03 15 19 02 01 04 70
```

```
Pre order : 12 10 3 2 1 4 11 15 19 70
```

Java Program to Find Whether a Path Exists Between 2 Given Nodes

[« Prev](#)

[Next »](#)

This is a java program to construct a undirected graph and check whether path exists between two vertices, if it exists class return true, false otherwise. Class implements standard Breadth First Search algorithm to traverse from given source node to destination node.

Here is the source code of the Java Program to Find Whether a Path Exists Between 2 Given Nodes. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a sample program to check that there exists a path between source node and destination node
2. import java.util.HashMap;
3. import java.util.LinkedHashSet;
4. import java.util.LinkedList;
5. import java.util.Map;
6. import java.util.Scanner;
7. import java.util.Set;
8.
9. public class Path_Between_Nodes
10. {
11.     private Map<String, LinkedHashSet<String>> map = new HashMap();
12.
13.     public void addEdge(String node1, String node2)
14.     {
15.         LinkedHashSet<String> adjacent = map.get(node1);
16.         if (adjacent == null)
17.         {
18.             adjacent = new LinkedHashSet();
19.             map.put(node1, adjacent);
20.         }
21.         adjacent.add(node2);
22.     }
23.
24.     public void addTwoWayVertex(String node1, String node2)
25.     {
26.         addEdge(node1, node2);
27.         addEdge(node2, node1);
28.     }
29.
30.     public boolean isConnected(String node1, String node2)
31.     {
32.         Set adjacent = map.get(node1);
33.         if (adjacent == null)
34.         {
35.             return false;
36.         }
37.         return adjacent.contains(node2);
38.     }
39.
40.     public LinkedList<String> adjacentNodes(String last)
41.     {
42.         LinkedHashSet<String> adjacent = map.get(last);
43.         if (adjacent == null)
44.         {
45.             return new LinkedList();
46.         }
47.         return new LinkedList<String>(adjacent);
48.     }
49.
50.     private static String START;
```

```

51. private static String END;
52. private static boolean flag;
53.
54. public static void main(String[] args)
55. {
56.     // this graph is directional
57.     Path_Between_Nodes graph = new Path_Between_Nodes();
58.     // graph.addEdge("A", "B");
59.     graph.addEdge("A", "C");
60.     graph.addEdge("B", "A");
61.     graph.addEdge("B", "D");
62.     graph.addEdge("B", "E");
63.     graph.addEdge("B", "F");
64.     graph.addEdge("C", "A");
65.     graph.addEdge("C", "E");
66.     graph.addEdge("C", "F");
67.     graph.addEdge("D", "B");
68.     graph.addEdge("E", "C");
69.     graph.addEdge("E", "F");
70.     // graph.addEdge("F", "B");
71.     graph.addEdge("F", "C");
72.     graph.addEdge("F", "E");
73.     LinkedList<String> visited = new LinkedList();
74.     System.out.println("Enter the source node:");
75.     Scanner sc = new Scanner(System.in);
76.     START = sc.next();
77.     System.out.println("Enter the destination node:");
78.     END = sc.next();
79.
80.     visited.add(START);
81.     new Path_Between_Nodes().breadthFirst(graph, visited);
82.     sc.close();
83. }
84.
85. private void breadthFirst(Path_Between_Nodes graph,
86.     LinkedList<String> visited)
87. {
88.     LinkedList<String> nodes = graph.adjacentNodes(visited.getLast());
89.
90.     for (String node : nodes)
91.     {
92.         if (visited.contains(node))
93.         {
94.             continue;
95.         }
96.         if (node.equals(END))
97.         {
98.             visited.add(node);
99.             printPath(visited);
100.            flag = true;
101.            visited.removeLast();
102.            break;
103.        }
104.    }
105.
106.    for (String node : nodes)
107.    {
108.        if (visited.contains(node) || node.equals(END))
109.        {
110.            continue;
111.        }
112.        visited.addLast(node);
113.        breadthFirst(graph, visited);
114.        visited.removeLast();
115.    }
116.    if (flag == false)

```

```

117.    {
118.        System.out.println("No path Exists between " + START + " and "
119.                            + END);
120.        flag = true;
121.    }
122.
123. }
124.
125. private void printPath(LinkedList<String> visited)
126. {
127.     if (flag == false)
128.         System.out.println("Yes there exists a path between " + START
129.                             + " and " + END);
130.
131.     for (String node : visited)
132.     {
133.         System.out.print(node);
134.         System.out.print(" ");
135.     }
136.     System.out.println();
137. }
138.

```

Output:

advertisement

```
$ javac Path_Between_Nodes.java
$ java Path_Between_Nodes
```

Enter the source node:

E

Enter the destination node:

B

No path Exists between E and B

Enter the source node:

A

Enter the destination node:

E

Yes there exists a path between A and E

A C E

A C F E

Java Program to Implement Interval Tree

This is a java program to implement Interval Tree. In computer science, an interval tree is an ordered tree data structure to hold intervals. Specifically, it allows one to efficiently find all intervals that overlap with any given interval or point. It is often used for windowing queries, for instance, to find all roads on a computerized map inside a rectangular viewport, or to find all visible elements inside a three-dimensional scene. A similar data structure is the segment tree. The trivial solution is to visit each interval and test whether it intersects the given point or interval, which requires $T(n)$ time, where n is the number of intervals in the collection.

Here is the source code of the Java Program to Implement Interval Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement a interval tree
2. import java.util.ArrayList;
3. import java.util.List;
4. import java.util.Map.Entry;
5. import java.util.SortedMap;
6. import java.util.SortedSet;
7. import java.util.TreeMap;
8. import java.util.TreeSet;
9.
10.class Interval<Type> implements Comparable<Interval<Type>>
11.
12.
13. private long start;
14. private long end;
15. private Type data;
16.
17. public Interval(long start, long end, Type data)
18. {
19.     this.start = start;
20.     this.end = end;
21.     this.data = data;
22. }
23.
24. public long getStart()
25. {
26.     return start;
27. }
28.
29. public void setStart(long start)
30. {
31.     this.start = start;
32. }
33.
34. public long getEnd()
35. {
36.     return end;
37. }
38.
39. public void setEnd(long end)
40. {
41.     this.end = end;
42. }
43.
44. public Type getData()
45. {
46.     return data;
47. }
48.
49. public void setData(Type data)
50. {
51.     this.data = data;
```

```

52. }
53.
54. public boolean contains(long time)
55. {
56.     return time < end && time > start;
57. }
58.
59. public boolean intersects(Interval<?> other)
60. {
61.     return other.getEnd() > start && other.getStart() < end;
62. }
63.
64. public int compareTo(Interval<Type> other)
65. {
66.     if (start < other.getStart())
67.         return -1;
68.     else if (start > other.getStart())
69.         return 1;
70.     else if (end < other.getEnd())
71.         return -1;
72.     else if (end > other.getEnd())
73.         return 1;
74.     else
75.         return 0;
76. }
77.
78. }
79.
80.class IntervalNode<Type>
81.
82.
83. private SortedMap<Interval<Type>, List<Interval<Type>>> intervals;
84. private long center;
85. private IntervalNode<Type> leftNode;
86. private IntervalNode<Type> rightNode;
87.
88. public IntervalNode()
89. {
90.     intervals = new TreeMap<Interval<Type>, List<Interval<Type>>>();
91.     center = 0;
92.     leftNode = null;
93.     rightNode = null;
94. }
95.
96. public IntervalNode(List<Interval<Type>> intervalList)
97. {
98.
99.     intervals = new TreeMap<Interval<Type>, List<Interval<Type>>>();
100.
101.    SortedSet<Long> endpoints = new TreeSet<Long>();
102.
103.    for (Interval<Type> interval : intervalList)
104.    {
105.        endpoints.add(interval.getStart());
106.        endpoints.add(interval.getEnd());
107.    }
108.
109.    long median = getMedian(endpoints);
110.    center = median;
111.
112.    List<Interval<Type>> left = new ArrayList<Interval<Type>>();
113.    List<Interval<Type>> right = new ArrayList<Interval<Type>>();
114.
115.    for (Interval<Type> interval : intervalList)
116.    {
117.        if (interval.getEnd() < median)

```

```

118.         left.add(interval);
119.     else if (interval.getStart() > median)
120.         right.add(interval);
121.     else
122.     {
123.         List<Interval<Type>> posting = intervals.get(interval);
124.         if (posting == null)
125.         {
126.             posting = new ArrayList<Interval<Type>>();
127.             intervals.put(interval, posting);
128.         }
129.         posting.add(interval);
130.     }
131. }
132.
133. if (left.size() > 0)
134.     leftNode = new IntervalNode<Type>(left);
135. if (right.size() > 0)
136.     rightNode = new IntervalNode<Type>(right);
137. }
138.
139. public List<Interval<Type>> stab(long time)
140. {
141.     List<Interval<Type>> result = new ArrayList<Interval<Type>>();
142.
143.     for (Entry<Interval<Type>, List<Interval<Type>>> entry : intervals
144.          .entrySet())
145.     {
146.         if (entry.getKey().contains(time))
147.             for (Interval<Type> interval : entry.getValue())
148.                 result.add(interval);
149.         else if (entry.getKey().getStart() > time)
150.             break;
151.     }
152.
153.     if (time < center && leftNode != null)
154.         result.addAll(leftNode.stab(time));
155.     else if (time > center && rightNode != null)
156.         result.addAll(rightNode.stab(time));
157.     return result;
158. }
159.
160. public List<Interval<Type>> query(Interval<?> target)
161. {
162.     List<Interval<Type>> result = new ArrayList<Interval<Type>>();
163.
164.     for (Entry<Interval<Type>, List<Interval<Type>>> entry : intervals
165.          .entrySet())
166.     {
167.         if (entry.getKey().intersects(target))
168.             for (Interval<Type> interval : entry.getValue())
169.                 result.add(interval);
170.         else if (entry.getKey().getStart() > target.getEnd())
171.             break;
172.     }
173.
174.     if (target.getStart() < center && leftNode != null)
175.         result.addAll(leftNode.query(target));
176.     if (target.getEnd() > center && rightNode != null)
177.         result.addAll(rightNode.query(target));
178.     return result;
179. }
180.
181. public long getCenter()
182. {
183.     return center;

```

```

184. }
185.
186. public void setCenter(long center)
187. {
188.     this.center = center;
189. }
190.
191. public IntervalNode<Type> getLeft()
192. {
193.     return leftNode;
194. }
195.
196. public void setLeft(IntervalNode<Type> left)
197. {
198.     this.leftNode = left;
199. }
200.
201. public IntervalNode<Type> getRight()
202. {
203.     return rightNode;
204. }
205.
206. public void setRight(IntervalNode<Type> right)
207. {
208.     this.rightNode = right;
209. }
210.
211. private Long getMedian(SortedSet<Long> set)
212. {
213.     int i = 0;
214.     int middle = set.size() / 2;
215.     for (Long point : set)
216.     {
217.         if (i == middle)
218.             return point;
219.         i++;
220.     }
221.     return null;
222. }
223.
224. @Override
225. public String toString()
226. {
227.     StringBuffer sb = new StringBuffer();
228.     sb.append(center + ": ");
229.     for (Entry<Interval<Type>, List<Interval<Type>>> entry : intervals
230.          .entrySet())
231.     {
232.         sb.append("[" + entry.getKey().getStart() + ","
233.                 + entry.getKey().getEnd() + "]:" );
234.         for (Interval<Type> interval : entry.getValue())
235.         {
236.             sb.append("(" + interval.getStart() + "," + interval.getEnd()
237.                     + "," + interval.getData() + ")");
238.         }
239.         sb.append("} ");
240.     }
241.     return sb.toString();
242. }
243.
244. }
245.
246. class IntervalTree<Type>
247. {
248.
249.     private IntervalNode<Type> head;

```

```

250. private List<Interval<Type>> intervalList;
251. private boolean inSync;
252. private int size;
253.
254. public IntervalTree()
255. {
256.     this.head = new IntervalNode<Type>();
257.     this.intervalList = new ArrayList<Interval<Type>>();
258.     this.inSync = true;
259.     this.size = 0;
260. }
261.
262. public IntervalTree(List<Interval<Type>> intervalList)
263. {
264.     this.head = new IntervalNode<Type>(intervalList);
265.     this.intervalList = new ArrayList<Interval<Type>>();
266.     this.intervalList.addAll(intervalList);
267.     this.inSync = true;
268.     this.size = intervalList.size();
269. }
270.
271. public List<Type> get(long time)
272. {
273.     List<Interval<Type>> intervals = getIntervals(time);
274.     List<Type> result = new ArrayList<Type>();
275.     for (Interval<Type> interval : intervals)
276.         result.add(interval.getData());
277.     return result;
278. }
279.
280. public List<Interval<Type>> getIntervals(long time)
281. {
282.     build();
283.     return head.stab(time);
284. }
285.
286. public List<Type> get(long start, long end)
287. {
288.     List<Interval<Type>> intervals = getIntervals(start, end);
289.     List<Type> result = new ArrayList<Type>();
290.     for (Interval<Type> interval : intervals)
291.         result.add(interval.getData());
292.     return result;
293. }
294.
295. public List<Interval<Type>> getIntervals(long start, long end)
296. {
297.     build();
298.     return head.query(new Interval<Type>(start, end, null));
299. }
300.
301. public void addInterval(Interval<Type> interval)
302. {
303.     intervalList.add(interval);
304.     inSync = false;
305. }
306.
307. public void addInterval(long begin, long end, Type data)
308. {
309.     intervalList.add(new Interval<Type>(begin, end, data));
310.     inSync = false;
311. }
312.
313. public boolean inSync()
314. {
315.     return inSync;

```

```

316. }
317.
318. public void build()
319. {
320.     if (!inSync)
321.     {
322.         head = new IntervalNode<Type>(intervalList);
323.         inSync = true;
324.         size = intervalList.size();
325.     }
326. }
327.
328. public int currentSize()
329. {
330.     return size;
331. }
332.
333. public int listSize()
334. {
335.     return intervalList.size();
336. }
337.
338. @Override
339. public String toString()
340. {
341.     return nodeString(head, 0);
342. }
343.
344. private String nodeString(IntervalNode<Type> node, int level)
345. {
346.     if (node == null)
347.         return "";
348.
349.     StringBuffer sb = new StringBuffer();
350.     for (int i = 0; i < level; i++)
351.         sb.append("\t");
352.     sb.append(node + "\n");
353.     sb.append(nodeString(node.getLeft(), level + 1));
354.     sb.append(nodeString(node.getRight(), level + 1));
355.     return sb.toString();
356. }
357. }
358.
359. public class IntervalTreeProblem
360. {
361.     public static void main(String args[])
362.     {
363.         IntervalTree<Integer> it = new IntervalTree<Integer>();
364.
365.         it.addInterval(0L, 10L, 1);
366.         it.addInterval(20L, 30L, 2);
367.         it.addInterval(15L, 17L, 3);
368.         it.addInterval(25L, 35L, 4);
369.
370.         List<Integer> result1 = it.get(5L);
371.         List<Integer> result2 = it.get(10L);
372.         List<Integer> result3 = it.get(29L);
373.         List<Integer> result4 = it.get(5L, 15L);
374.
375.         System.out.print("\nIntervals that contain 5L:");
376.         for (int r : result1)
377.             System.out.print(r + " ");
378.
379.         System.out.print("\nIntervals that contain 10L:");
380.         for (int r : result2)
381.             System.out.print(r + " ");

```

```
382.  
383.     System.out.print("\nIntervals that contain 29L:");  
384.     for (int r : result3)  
385.         System.out.print(r + " ");  
386.  
387.     System.out.print("\nIntervals that intersect (5L,15L):");  
388.     for (int r : result4)  
389.         System.out.print(r + " ");  
390.     }  
391. }
```

Output:

advertisement

```
$ javac IntervalTreeImplementation.java  
$ java IntervalTreeImplementation
```

Interval Tree Implementation

```
Intervals that contain 5L:1  
Intervals that contain 10L:  
Intervals that contain 29L:2 4  
Intervals that intersect (5L,15L):1
```

260.Java Program to Implement Range Tree

[« Prev](#)

[Next »](#)

This is a java program to implement Range Tree. A range tree on a set of 1-dimensional points is a balanced binary search tree on those points. The points stored in the tree are stored in the leaves of the tree; each internal node stores the largest value contained in its left subtree. A range tree on a set of points in d-dimensions is a recursively defined multi-level binary search tree. Each level of the data structure is a binary search tree on one of the d-dimensions. The first level is a binary search tree on the first of the d-coordinates. Each vertex v of this tree contains an associated structure that is a (d-1)-dimensional range tree on the last (d-1)-coordinates of the points stored in the subtree of v.

Here is the source code of the Java Program to Implement Range Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement Range Tree
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. class BSTNodes
6. {
7.     BSTNodes left, right;
8.     int data;
9.
10.    public BSTNodes()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.    }
16.
17.    public BSTNodes(int n)
18.    {
19.        left = null;
20.        right = null;
21.        data = n;
22.    }
23.
24.    public void setLeft(BSTNodes n)
25.    {
26.        left = n;
27.    }
28.
29.    public void setRight(BSTNodes n)
30.    {
31.        right = n;
32.    }
33.
34.    public BSTNodes getLeft()
35.    {
36.        return left;
37.    }
38.
39.    public BSTNodes getRight()
40.    {
41.        return right;
42.    }
43.
44.    public void setData(int d)
45.    {
46.        data = d;
47.    }
48.
```

```

49. public int getData()
50. {
51.     return data;
52. }
53.}
54.
55.class BST
56.{ 
57. private BSTNodes root;
58.
59. public BST()
60. {
61.     root = null;
62. }
63.
64. public boolean isEmpty()
65. {
66.     return root == null;
67. }
68.
69. public void insert(int data)
70. {
71.     root = insert(root, data);
72. }
73.
74. private BSTNodes insert(BSTNodes node, int data)
75. {
76.     if (node == null)
77.         node = new BSTNodes(data);
78.     else
79.     {
80.         if (data <= node.getData())
81.             node.left = insert(node.left, data);
82.         else
83.             node.right = insert(node.right, data);
84.     }
85.     return node;
86. }
87.
88. public void delete(int k)
89. {
90.     if (isEmpty())
91.         System.out.println("Tree Empty");
92.     else if (search(k) == false)
93.         System.out.println("Sorry " + k + " is not present");
94.     else
95.     {
96.         root = delete(root, k);
97.         System.out.println(k + " deleted from the tree");
98.     }
99. }
100.
101. private BSTNodes delete(BSTNodes root, int k)
102. {
103.     BSTNodes p, p2, n;
104.     if (root.getData() == k)
105.     {
106.         BSTNodes lt, rt;
107.         lt = root.getLeft();
108.         rt = root.getRight();
109.         if (lt == null && rt == null)
110.             return null;
111.         else if (lt == null)
112.         {
113.             p = rt;
114.             return p;

```

```

115.     } else if (rt == null)
116.     {
117.         p = lt;
118.         return p;
119.     } else
120.     {
121.         p2 = rt;
122.         p = rt;
123.         while (p.getLeft() != null)
124.             p = p.getLeft();
125.         p.setLeft(lt);
126.         return p2;
127.     }
128. }
129. if (k < root.getData())
130. {
131.     n = delete(root.getLeft(), k);
132.     root.setLeft(n);
133. } else
134. {
135.     n = delete(root.getRight(), k);
136.     root.setRight(n);
137. }
138. return root;
139. }
140.
141. public int countNodes()
142. {
143.     return countNodes(root);
144. }
145.
146. private int countNodes(BSTNodes r)
147. {
148.     if (r == null)
149.         return 0;
150.     else
151.     {
152.         int l = 1;
153.         l += countNodes(r.getLeft());
154.         l += countNodes(r.getRight());
155.         return l;
156.     }
157. }
158.
159. public boolean search(int val)
160. {
161.     return search(root, val);
162. }
163.
164. private boolean search(BSTNodes r, int val)
165. {
166.     boolean found = false;
167.     while ((r != null) && !found)
168.     {
169.         int rval = r.getData();
170.         if (val < rval)
171.             r = r.getLeft();
172.         else if (val > rval)
173.             r = r.getRight();
174.         else
175.         {
176.             found = true;
177.             break;
178.         }
179.         found = search(r, val);
180.     }

```

```

181.     return found;
182. }
183.
184. public void inorder()
185. {
186.     inorder(root);
187. }
188.
189. private void inorder(BSTNodes r)
190. {
191.     if (r != null)
192.     {
193.         inorder(r.getLeft());
194.         System.out.print(r.getData() + " ");
195.         inorder(r.getRight());
196.     }
197. }
198.
199. public void preorder()
200. {
201.     preorder(root);
202. }
203.
204. private void preorder(BSTNodes r)
205. {
206.     if (r != null)
207.     {
208.         System.out.print(r.getData() + " ");
209.         preorder(r.getLeft());
210.         preorder(r.getRight());
211.     }
212. }
213.
214. public void postorder()
215. {
216.     postorder(root);
217. }
218.
219. private void postorder(BSTNodes r)
220. {
221.     if (r != null)
222.     {
223.         postorder(r.getLeft());
224.         postorder(r.getRight());
225.         System.out.print(r.getData() + " ");
226.     }
227. }
228. }
229.
230. public class RangeTree
231. {
232.     public static void main(String[] args)
233.     {
234.         Scanner scan = new Scanner(System.in);
235.         BST bst = new BST();
236.         System.out
237.             .println("Range Tree in One Dimensional points(Binary Search Tree)\n");
238.         Random random = new Random();
239.         int N = 10;
240.         for (int i = 0; i < N; i++)
241.             bst.insert(Math.abs(random.nextInt(100)));
242.
243.         char ch;
244.         do
245.         {
246.             System.out.print("Operations\n");

```

```

247.     System.out.println("1. Print Tree ");
248.     System.out.println("2. Delete");
249.     System.out.println("3. Search");
250.     System.out.println("4. Count Nodes");
251.     System.out.println("5. Check Empty");
252.
253.     int choice = scan.nextInt();
254.     switch (choice)
255.     {
256.         case 1:
257.             System.out.print("\nPost order : ");
258.             bst.postorder();
259.             System.out.print("\nPre order : ");
260.             bst.preorder();
261.             System.out.print("\nIn order : ");
262.             bst.inorder();
263.             break;
264.         case 2:
265.             System.out.println("Enter integer element to delete");
266.             bst.delete(scan.nextInt());
267.             break;
268.         case 3:
269.             System.out.println("Enter integer element to search");
270.             System.out.println("Search result : "
271.                 + bst.search(scan.nextInt()));
272.             break;
273.         case 4:
274.             System.out.println("Nodes = " + bst.countNodes());
275.             break;
276.         case 5:
277.             System.out.println("Empty status = " + bst.isEmpty());
278.             break;
279.         default:
280.             System.out.println("Wrong Entry \n ");
281.             break;
282.     }
283.
284.     System.out.println("\nDo you want to continue (Type y or n) \n");
285.     ch = scan.next().charAt(0);
286. } while (ch == 'Y' || ch == 'y');
287. scan.close();
288. }
289. }
```

Output:

advertisement

```
$ javac RangeTree.java
$ java RangeTree
```

Range Tree in One Dimensional points(Binary Search Tree)

Operations

1. Print Tree
2. Delete
3. Search
4. Count Nodes
5. Check Empty

1

Post order : 15 14 12 29 49 47 22 92 91 7

Pre order : 7 91 22 12 14 15 47 29 49 92

In order : 7 12 14 15 22 29 47 49 91 92

Do you want to continue (Type y or n)

y

Operations
1. Print Tree
2. Delete
3. Search
4. Count Nodes
5. Check Empty
2
Enter integer element to delete
7
7 deleted from the tree

Do you want to continue (Type y or n)

y
Operations
1. Print Tree
2. Delete
3. Search
4. Count Nodes
5. Check Empty
3
Enter integer element to search
91
Search result : true

Do you want to continue (Type y or n)

y
Operations
1. Print Tree
2. Delete
3. Search
4. Count Nodes
5. Check Empty
4
Nodes = 9

Do you want to continue (Type y or n)

y
Operations
1. Print Tree
2. Delete
3. Search
4. Count Nodes
5. Check Empty
5
Empty status = false

Do you want to continue (Type y or n)

n

261.Java Program to Find the Shortest Path from Source Vertex to All Other Vertices in Linear Time

[« Prev](#)

[Next »](#)

This Java program is to Implement weighted graph and find shortest path from one source vertex to every other vertex. Dijkstra's Algorithm can be used to achieve this goal.

Here is the source code of the Java Program to Find the Shortest Path from Source Vertex to All Other Vertices in Linear Time. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the shortest path from one vertex to all other vertex
2. import java.util.HashSet;
3. import java.util.InputMismatchException;
4. import java.util.Iterator;
5. import java.util.Scanner;
6. import java.util.Set;
7.
8. public class Shortest_Path_to_AllVertex
9. {
10.     private int      distances[];
11.     private Set<Integer> settled;
12.     private Set<Integer> unsettled;
13.     private int      number_of_nodes;
14.     private int      adjacencyMatrix[][];
15.
16.     public Shortest_Path_to_AllVertex(int number_of_nodes)
17.     {
18.         this.number_of_nodes = number_of_nodes;
19.         distances = new int[number_of_nodes + 1];
20.         settled = new HashSet<Integer>();
21.         unsettled = new HashSet<Integer>();
22.         adjacencyMatrix = new int[number_of_nodes + 1][number_of_nodes + 1];
23.     }
24.
25.     public void shortestPath(int adjacency_matrix[][], int source)
26.     {
27.         int evaluationNode;
28.         for (int i = 1; i <= number_of_nodes; i++)
29.             for (int j = 1; j <= number_of_nodes; j++)
30.                 adjacencyMatrix[i][j] = adjacency_matrix[i][j];
31.
32.         for (int i = 1; i <= number_of_nodes; i++)
33.         {
34.             distances[i] = Integer.MAX_VALUE;
35.         }
36.
37.         unsettled.add(source);
38.         distances[source] = 0;
39.         while (!unsettled.isEmpty())
40.         {
41.             evaluationNode = getNodeWithMinimumDistanceFromUnsettled();
42.             unsettled.remove(evaluationNode);
43.             settled.add(evaluationNode);
44.             evaluateNeighbours(evaluationNode);
45.         }
46.     }
47.
48.     private int getNodeWithMinimumDistanceFromUnsettled()
```

```

49. {
50.     int min;
51.     int node = 0;
52.
53.     Iterator<Integer> iterator = unsettled.iterator();
54.     node = iterator.next();
55.     min = distances[node];
56.     for (int i = 1; i <= distances.length; i++)
57.     {
58.         if (unsettled.contains(i))
59.         {
60.             if (distances[i] <= min)
61.             {
62.                 min = distances[i];
63.                 node = i;
64.             }
65.         }
66.     }
67.     return node;
68. }
69.
70. private void evaluateNeighbours(int evaluationNode)
71. {
72.     int edgeDistance = -1;
73.     int newDistance = -1;
74.
75.     for (int destinationNode = 1; destinationNode <= number_of_nodes; destinationNode++)
76.     {
77.         if (!settled.contains(destinationNode))
78.         {
79.             if (adjacencyMatrix[evaluationNode][destinationNode] != Integer.MAX_VALUE)
80.             {
81.                 edgeDistance = adjacencyMatrix[evaluationNode][destinationNode];
82.                 newDistance = distances[evaluationNode] + edgeDistance;
83.                 if (newDistance < distances[destinationNode])
84.                 {
85.                     distances[destinationNode] = newDistance;
86.                 }
87.                 unsettled.add(destinationNode);
88.             }
89.         }
90.     }
91. }
92.
93. public static void main(String... arg)
94. {
95.     int adjacency_matrix[][];
96.     int number_of_vertices;
97.     int source = 0;
98.     Scanner scan = new Scanner(System.in);
99.     try
100.    {
101.        System.out.println("Enter the number of vertices");
102.        number_of_vertices = scan.nextInt();
103.        adjacency_matrix = new int[number_of_vertices + 1][number_of_vertices + 1];
104.
105.        System.out.println("Enter the Weighted Matrix for the graph");
106.        for (int i = 1; i <= number_of_vertices; i++)
107.        {
108.            for (int j = 1; j <= number_of_vertices; j++)
109.            {
110.                adjacency_matrix[i][j] = scan.nextInt();
111.                if (i == j)
112.                {
113.                    adjacency_matrix[i][j] = 0;
114.                    continue;

```

```

115.        }
116.        if (adjacency_matrix[i][j] == 0)
117.        {
118.            adjacency_matrix[i][j] = Integer.MAX_VALUE;
119.        }
120.    }
121.}
122.
123. System.out.println("Enter the source ");
124. source = scan.nextInt();
125.
126. Shortest_Path_to_AllVertex sp = new Shortest_Path_to_AllVertex(
127.     number_of_vertices);
128. sp.shortestPath(adjacency_matrix, source);
129.
130. System.out.println("The Shorted Path from " + source
131.         + " to all other nodes are:");
132. for (int i = 1; i <= sp.distances.length - 1; i++)
133. {
134.
135.     System.out.println(source + " to " + i + " is "
136.         + sp.distances[i]);
137. }
138. } catch (InputMismatchException inputMismatch)
139. {
140.     System.out.println("Wrong Input Format");
141. }
142. scan.close();
143. }
144.

```

Output:

advertisement

```
$ javac Shortest_Path_to_AllVertex.java
$ java Shortest_Path_to_AllVertex
```

Enter the number of vertices

5

Enter the Weighted Matrix for the graph

```
0 9 6 5 3
0 0 0 0 0
0 2 0 4 0
0 0 0 0 0
0 0 0 0 0
```

Enter the source

1

The Shorted Path from 1 to all other nodes are:

```
1 to 1 is 0
1 to 2 is 8
1 to 3 is 6
1 to 4 is 5
1 to 5 is 3
```

Java Program to Check Whether an Input Binary Tree is the Sub Tree of the Binary Tree

This Java program is to Implement binary tree and check whether a tree is subtree of another. This can be done in two ways. A tree can be subtree of another if they have same structure (same object references but different value) and with same structure and values. This given class checks for both.

Here is the source code of the Java Program to Check Whether an Input Binary Tree is the Sub Tree of the Binary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether a binary tree is subtree of another tree
2. class Btrees
3. {
4.     Object value;
5.     Btrees Left;
6.     Btrees Right;
7.
8.     Btrees(int k)
9.     {
10.         value = k;
11.     }
12. }
13.
14. public class SubBinaryTree
15. {
16.     public static boolean ifsubtreeRef(Btrees t1, Btrees t2)
17.     {
18.         if (t2 == null)
19.             return true;
20.         if (t1 == null)
21.             return false;
22.         return (t1 == t2) || ifsubtreeRef(t1.Left, t2)
23.             || ifsubtreeRef(t1.Right, t2);
24.     }
25.
26.     public static boolean ifsubtreeValue(Btrees t1, Btrees t2)
27.     {
28.         if (t2 == null)
29.             return true;
30.         if (t1 == null)
31.             return false;
32.         if (t1.value == t2.value)
33.             if (ifsubtreeValue(t1.Left, t2.Left)
34.                 && ifsubtreeValue(t1.Right, t2.Right))
35.                 return true;
36.         return ifsubtreeValue(t1.Left, t2) || ifsubtreeValue(t1.Right, t2);
37.     }
38.
39.     public static void main(String[] args)
40.     {
41.         Btrees t1 = new Btrees(1);
42.         t1.Left = new Btrees(2);
43.         t1.Right = new Btrees(3);
44.         t1.Right.Left = new Btrees(4);
45.         t1.Right.Right = new Btrees(5);
46.
47.         Btrees t2 = new Btrees(3);
48.         t2.Left = new Btrees(4);
49.         t2.Right = new Btrees(5);
50.
51.         if (ifsubtreeRef(t1, t2))
52.             System.out.println("T2 is sub-tree of T1 (Reference wise)");
53.         else
54.             System.out.println("T2 is NOT sub-tree of T1 (Reference wise);
```

```
55.  
56. if (ifsubtreeValue(t1, t2))  
57.     System.out.println("T2 is sub-tree of T1 (Value wise);  
58. else  
59.     System.out.println("T2 is NOT sub-tree of T1 (Value wise)");  
60. }  
61.}
```

Output:

advertisement

```
$ javac SubBinaryTree.java  
$ java SubBinaryTree
```

T2 is NOT sub-tree of T1 (Reference wise)
T2 is sub-tree of T1 (Value wise)

262.Java Program to Implement Ternary Tree

[« Prev](#)

[Next »](#)

This Java program is to Implement ternary tree. In computer science, a ternary tree is a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”. Nodes with children are parent nodes, and child nodes may contain references to their parents. Outside the tree, there is often a reference to the “root” node (the ancestor of all nodes), if it exists. Any node in the data structure can be reached by starting at root node and repeatedly following references to either the left, mid or right child.

Here is the source code of the Java Program to Implement Ternary Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement Ternary Tree
2. import java.util.Scanner;
3. import java.util.ArrayList;
4.
5. class TSTNode
6. {
7.     char data;
8.     boolean isEnd;
9.     TSTNode left, middle, right;
10.
11.    public TSTNode(char data)
12.    {
13.        this.data = data;
14.        this.isEnd = false;
15.        this.left = null;
16.        this.middle = null;
17.        this.right = null;
18.    }
19.
20.
21. class TernarySearchTree
22. {
23.     private TSTNode root;
24.     private ArrayList<String> al;
25.
26.     public TernarySearchTree()
27.     {
28.         root = null;
29.     }
30.
31.     public boolean isEmpty()
32.     {
33.         return root == null;
34.     }
35.
36.     public void makeEmpty()
37.     {
38.         root = null;
39.     }
40.
41.     public void insert(String word)
42.     {
43.         root = insert(root, word.toCharArray(), 0);
44.     }
45.
46.     public TSTNode insert(TSTNode r, char[] word, int ptr)
47.     {
48.         if (r == null)
49.             r = new TSTNode(word[ptr]);
50.         if (word[ptr] < r.data)
```

```

51.     r.left = insert(r.left, word, ptr);
52.   else if (word[ptr] > r.data)
53.     r.right = insert(r.right, word, ptr);
54.   else
55.   {
56.     if (ptr + 1 < word.length)
57.       r.middle = insert(r.middle, word, ptr + 1);
58.     else
59.       r.isEnd = true;
60.   }
61.   return r;
62. }
63.
64. public void delete(String word)
65. {
66.   delete(root, word.toCharArray(), 0);
67. }
68.
69. private void delete(TSTNode r, char[] word, int ptr)
70. {
71.   if (r == null)
72.     return;
73.   if (word[ptr] < r.data)
74.     delete(r.left, word, ptr);
75.   else if (word[ptr] > r.data)
76.     delete(r.right, word, ptr);
77.   else
78.   {
79.
80.     if (r.isEnd && ptr == word.length - 1)
81.       r.isEnd = false;
82.     else if (ptr + 1 < word.length)
83.       delete(r.middle, word, ptr + 1);
84.   }
85. }
86.
87. public boolean search(String word)
88. {
89.   return search(root, word.toCharArray(), 0);
90. }
91.
92. private boolean search(TSTNode r, char[] word, int ptr)
93. {
94.   if (r == null)
95.     return false;
96.   if (word[ptr] < r.data)
97.     return search(r.left, word, ptr);
98.   else if (word[ptr] > r.data)
99.     return search(r.right, word, ptr);
100.  else
101.  {
102.    if (r.isEnd && ptr == word.length - 1)
103.      return true;
104.    else if (ptr == word.length - 1)
105.      return false;
106.    else
107.      return search(r.middle, word, ptr + 1);
108.  }
109. }
110.
111. public String toString()
112. {
113.   al = new ArrayList<String>();
114.   traverse(root, "");
115.   return "\nTernary Search Tree : " + al;
116. }

```

```

117.
118. private void traverse(TSTNode r, String str)
119. {
120.     if (r != null)
121.     {
122.         traverse(r.left, str);
123.         str = str + r.data;
124.         if (r.isEnd)
125.             al.add(str);
126.         traverse(r.middle, str);
127.         str = str.substring(0, str.length() - 1);
128.         traverse(r.right, str);
129.     }
130. }
131.}
132.
133.public class TernaryTree
134.{
135.    public static void main(String[] args)
136.    {
137.        Scanner scan = new Scanner(System.in);
138.
139.        TernarySearchTree tst = new TernarySearchTree();
140.        System.out.println("Ternary Search Tree Test\n");
141.        char ch;
142.
143.        do
144.        {
145.            System.out.println("\nTernary Search Tree Operations\n");
146.            System.out.println("1. insert word");
147.            System.out.println("2. search word");
148.            System.out.println("3. delete word");
149.            System.out.println("4. check empty");
150.            System.out.println("5. make empty");
151.            int choice = scan.nextInt();
152.            switch (choice)
153.            {
154.                case 1:
155.                    System.out.println("Enter word to insert");
156.                    tst.insert(scan.next());
157.                    break;
158.                case 2:
159.                    System.out.println("Enter word to search");
160.                    System.out.println("Search result : "
161.                        + tst.search(scan.next()));
162.                    break;
163.                case 3:
164.                    System.out.println("Enter word to delete");
165.                    tst.delete(scan.next());
166.                    break;
167.                case 4:
168.                    System.out.println("Empty Status : " + tst.isEmpty());
169.                    break;
170.                case 5:
171.                    System.out.println("Ternary Search Tree cleared");
172.                    tst.makeEmpty();
173.                    break;
174.                default:
175.                    System.out.println("Wrong Entry \n ");
176.                    break;
177.            }
178.            System.out.println(tst);
179.            System.out.println("\nDo you want to continue (Type y or n) \n");
180.            ch = scan.next().charAt(0);
181.        } while (ch == 'Y' || ch == 'y');
182.        scan.close();

```

183. }
184. }

Output:

advertisement

```
$ javac TernaryTree.java  
$ java TernaryTree
```

Ternary Tree Test

Ternary Search Tree Operations

1. insert word
2. search word
3. delete word
4. check empty
5. make empty

4

Empty Status : true

Ternary Search Tree : []

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word
3. delete word
4. check empty
5. make empty

1

Enter word to insert

Sanfoundry

Ternary Search Tree : [Sanfoundry]

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word
3. delete word
4. check empty
5. make empty

1

Enter word to insert

Technology

Ternary Search Tree : [Sanfoundry, Technology]

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word
3. delete word

4. check empty
5. make empty

1

Enter word to insert
Solutions

Ternary Search Tree : [Sanfoundry, Solutions, Technology]

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word

3. delete word

4. check empty

5. make empty

3

Enter word to delete
Solutions

Ternary Search Tree : [Sanfoundry, Technology]

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word

3. delete word

4. check empty

5. make empty

1

Enter word to insert

Blog

Ternary Search Tree : [Blog, Sanfoundry, Technology]

Do you want to continue (Type y or n)

y

Ternary Search Tree Operations

1. insert word
2. search word

3. delete word

4. check empty

5. make empty

2

Enter word to search

Sanfoundry

Search result : true

Ternary Search Tree : [Blog, Sanfoundry, Technology]

Do you want to continue (Type y or n)

n

263.Java Program to Perform Right Rotation on a Binary Search Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement Self Balancing Binary Search Tree. A self-balancing (or height-balanced) binary search tree is any node-based binary search tree that automatically keeps its height (maximal number of levels below the root) small in the face of arbitrary item insertions and deletions. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets. The implementation of self balancing binary search tree is similar to that of a AVL Tree data structure.

Here is the source code of the Java Program to Perform Right Rotation on a Binary Search Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement self balancinf binary search trees and indicate when right rotation is performed
2. import java.util.Scanner;
3.
4. class SBBST
5. {
6.     SBBST left, right;
7.     int data;
8.     int height;
9.
10.    public SBBST()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.        height = 0;
16.    }
17.
18.    public SBBST(int n)
19.    {
20.
21.        left = null;
22.        right = null;
23.        data = n;
24.        height = 0;
25.    }
26.}
27.
28.class SelfBalancingBinarySearchTree
29.
30. private SBBST root;
31.
32. public SelfBalancingBinarySearchTree()
33. {
34.     root = null;
35. }
36.
37. public boolean isEmpty()
38. {
39.     return root == null;
40. }
41.
42. public void clear()
43. {
44.     root = null;
45. }
46.
47. public void insert(int data)
```

```

48. {
49.     root = insert(data, root);
50. }
51.
52. private int height(SBBST t)
53. {
54.
55.     return t == null ? -1 : t.height;
56. }
57.
58. private int max(int lhs, int rhs)
59. {
60.     return lhs > rhs ? lhs : rhs;
61. }
62.
63. private SBBST insert(int x, SBBST t)
64. {
65.     if (t == null)
66.         t = new SBBST(x);
67.     else if (x < t.data)
68.     {
69.         t.left = insert(x, t.left);
70.         if (height(t.left) - height(t.right) == 2)
71.             if (x < t.left.data)
72.                 t = rotateWithLeftChild(t);
73.             else
74.                 t = doubleWithLeftChild(t);
75.     } else if (x > t.data)
76.     {
77.         t.right = insert(x, t.right);
78.         if (height(t.right) - height(t.left) == 2)
79.             if (x > t.right.data)
80.                 t = rotateWithRightChild(t);
81.             else
82.                 t = doubleWithRightChild(t);
83.     } else
84.     ;
85.     t.height = max(height(t.left), height(t.right)) + 1;
86.     return t;
87. }
88.
89. private SBBST rotateWithLeftChild(SBBST k2)
90. {
91.     //System.out.println("Left Rotation Performed");
92.     SBBST k1 = k2.left;
93.     k2.left = k1.right;
94.     k1.right = k2;
95.     k2.height = max(height(k2.left), height(k2.right)) + 1;
96.     k1.height = max(height(k1.left), k2.height) + 1;
97.     return k1;
98. }
99.
100. private SBBST rotateWithRightChild(SBBST k1)
101. {
102.     System.out.println("Right Rotation Performed");
103.     SBBST k2 = k1.right;
104.     k1.right = k2.left;
105.     k2.left = k1;
106.     k1.height = max(height(k1.left), height(k1.right)) + 1;
107.     k2.height = max(height(k2.right), k1.height) + 1;
108.     return k2;
109. }
110.
111. private SBBST doubleWithLeftChild(SBBST k3)
112. {
113.     //System.out.println("Left Rotation Performed");

```

```

114.     k3.left = rotateWithRightChild(k3.left);
115.     return rotateWithLeftChild(k3);
116. }
117.
118. private SBBST doubleWithRightChild(SBBST k1)
119. {
120.     System.out.println("Right Rotation Performed");
121.     k1.right = rotateWithLeftChild(k1.right);
122.     return rotateWithRightChild(k1);
123. }
124.
125. public int countNodes()
126. {
127.     return countNodes(root);
128. }
129.
130. private int countNodes(SBBST r)
131. {
132.     if (r == null)
133.         return 0;
134.     else
135.     {
136.         int l = 1;
137.         l += countNodes(r.left);
138.         l += countNodes(r.right);
139.         return l;
140.     }
141. }
142.
143. public boolean search(int val)
144. {
145.     return search(root, val);
146. }
147.
148. private boolean search(SBBST r, int val)
149. {
150.     boolean found = false;
151.     while ((r != null) && !found)
152.     {
153.         int rval = r.data;
154.         if (val < rval)
155.             r = r.left;
156.         else if (val > rval)
157.             r = r.right;
158.         else
159.         {
160.             found = true;
161.             break;
162.         }
163.         found = search(r, val);
164.     }
165.     return found;
166. }
167.
168. public void inorder()
169. {
170.     inorder(root);
171. }
172.
173. private void inorder(SBBST r)
174. {
175.     if (r != null)
176.     {
177.         inorder(r.left);
178.         System.out.print(r.data + " ");
179.         inorder(r.right);

```

```

180.     }
181. }
182.
183. public void preorder()
184. {
185.
186.     preorder(root);
187. }
188.
189. private void preorder(SBBST r)
190. {
191.     if (r != null)
192.     {
193.         System.out.print(r.data + " ");
194.         preorder(r.left);
195.         preorder(r.right);
196.     }
197. }
198.
199. public void postorder()
200. {
201.     postorder(root);
202. }
203.
204. private void postorder(SBBST r)
205. {
206.     if (r != null)
207.     {
208.         postorder(r.left);
209.         postorder(r.right);
210.         System.out.print(r.data + " ");
211.     }
212. }
213. }
214.
215. public class Right_Rotation_BST
216. {
217.     public static void main(String[] args)
218.     {
219.         Scanner scan = new Scanner(System.in);
220.
221.         SelfBalancingBinarySearchTree sbbst = new SelfBalancingBinarySearchTree();
222.         System.out.println("Self Balancing Tree\n");
223.
224.         System.out.println("Inset first 10 Elements");
225.         int N = 10;
226.         for (int i = 0; i < N; i++)
227.         {
228.             sbbst.insert(scan.nextInt());
229.
230.             System.out.println("\nPre-order :");
231.             sbbst.preorder();
232.             System.out.println("\nIn-order :");
233.             sbbst.inorder();
234.             System.out.println("\nPost-order :");
235.             sbbst.postorder();
236.
237.             System.out.println();
238.         }
239.         scan.close();
240.     }
241. }

```

Output:

advertisement

```
$ javac Right_Rotation_BST.java
$ java Right_Rotation_BST
```

Self Balancing Tree

Inset first 10 Elements

1

Pre-order :

1

In-order :

1

Post-order :

1

2

Pre-order :

1 2

In-order :

1 2

Post-order :

2 1

3

Right Rotation Performed

Pre-order :

2 1 3

In-order :

1 2 3

Post-order :

1 3 2

4

Pre-order :

2 1 3 4

In-order :

1 2 3 4

Post-order :

1 4 3 2

5

Right Rotation Performed

Pre-order :

2 1 4 3 5

In-order :

1 2 3 4 5

Post-order :

1 3 5 4 2

6

Right Rotation Performed

Pre-order :

4 2 1 3 5 6

In-order :

1 2 3 4 5 6

Post-order :

1 3 2 6 5 4

7

Right Rotation Performed

Pre-order :

4 2 1 3 6 5 7

In-order :

1 2 3 4 5 6 7

Post-order :

1 3 2 5 7 6 4

8

Pre-order :
4 2 1 3 6 5 7 8

In-order :
1 2 3 4 5 6 7 8
Post-order :
1 3 2 5 8 7 6 4

9

Right Rotation Performed

Pre-order :
4 2 1 3 6 5 8 7 9

In-order :
1 2 3 4 5 6 7 8 9
Post-order :
1 3 2 5 7 9 8 6 4

10

Right Rotation Performed

Pre-order :
4 2 1 3 8 6 5 7 9 10
In-order :
1 2 3 4 5 6 7 8 9 10
Post-order :
1 3 2 5 7 6 10 9 8 4

264.Java Program to Perform Left Rotation on a Binary Search Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement Self Balancing Binary Search Tree. A self-balancing (or height-balanced) binary search tree is any node-based binary search tree that automatically keeps its height (maximal number of levels below the root) small in the face of arbitrary item insertions and deletions. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets. The implementation of self balancing binary search tree is similar to that of a AVL Tree data structure.

Here is the source code of the Java Program to Perform Left Rotation on a Binary Search Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement self balancing binary search trees and indicate when left rotation is performed
2. import java.util.Scanner;
3.
4. class SBBST
5. {
6.     SBBST left, right;
7.     int data;
8.     int height;
9.
10.    public SBBST()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.        height = 0;
16.    }
17.
18.    public SBBST(int n)
19.    {
20.
21.        left = null;
22.        right = null;
23.        data = n;
24.        height = 0;
25.    }
26.}
27.
28.class SelfBalancingBinarySearchTree
29.{
30.    private SBBST root;
31.
32.    public SelfBalancingBinarySearchTree()
33.    {
34.        root = null;
35.    }
36.
37.    public boolean isEmpty()
38.    {
39.        return root == null;
40.    }
41.
42.    public void clear()
43.    {
44.        root = null;
45.    }
46.
47.    public void insert(int data)
```

```

48. {
49.     root = insert(data, root);
50. }
51.
52. private int height(SBBST t)
53. {
54.
55.     return t == null ? -1 : t.height;
56. }
57.
58. private int max(int lhs, int rhs)
59. {
60.     return lhs > rhs ? lhs : rhs;
61. }
62.
63. private SBBST insert(int x, SBBST t)
64. {
65.     if (t == null)
66.         t = new SBBST(x);
67.     else if (x < t.data)
68.     {
69.         t.left = insert(x, t.left);
70.         if (height(t.left) - height(t.right) == 2)
71.             if (x < t.left.data)
72.                 t = rotateWithLeftChild(t);
73.             else
74.                 t = doubleWithLeftChild(t);
75.     } else if (x > t.data)
76.     {
77.         t.right = insert(x, t.right);
78.         if (height(t.right) - height(t.left) == 2)
79.             if (x > t.right.data)
80.                 t = rotateWithRightChild(t);
81.             else
82.                 t = doubleWithRightChild(t);
83.     } else
84.     ;
85.     t.height = max(height(t.left), height(t.right)) + 1;
86.     return t;
87. }
88.
89. private SBBST rotateWithLeftChild(SBBST k2)
90. {
91.     System.out.println("Left Rotation Performed");
92.     SBBST k1 = k2.left;
93.     k2.left = k1.right;
94.     k1.right = k2;
95.     k2.height = max(height(k2.left), height(k2.right)) + 1;
96.     k1.height = max(height(k1.left), k2.height) + 1;
97.     return k1;
98. }
99.
100. private SBBST rotateWithRightChild(SBBST k1)
101. {
102.     //System.out.println("Right Rotation Performed");
103.     SBBST k2 = k1.right;
104.     k1.right = k2.left;
105.     k2.left = k1;
106.     k1.height = max(height(k1.left), height(k1.right)) + 1;
107.     k2.height = max(height(k2.right), k1.height) + 1;
108.     return k2;
109. }
110.
111. private SBBST doubleWithLeftChild(SBBST k3)
112. {
113.     System.out.println("Left Rotation Performed");

```

```

114.     k3.left = rotateWithRightChild(k3.left);
115.     return rotateWithLeftChild(k3);
116. }
117.
118. private SBBST doubleWithRightChild(SBBST k1)
119. {
120.     //System.out.println("Right Rotation Performed");
121.     k1.right = rotateWithLeftChild(k1.right);
122.     return rotateWithRightChild(k1);
123. }
124.
125. public int countNodes()
126. {
127.     return countNodes(root);
128. }
129.
130. private int countNodes(SBBST r)
131. {
132.     if (r == null)
133.         return 0;
134.     else
135.     {
136.         int l = 1;
137.         l += countNodes(r.left);
138.         l += countNodes(r.right);
139.         return l;
140.     }
141. }
142.
143. public boolean search(int val)
144. {
145.     return search(root, val);
146. }
147.
148. private boolean search(SBBST r, int val)
149. {
150.     boolean found = false;
151.     while ((r != null) && !found)
152.     {
153.         int rval = r.data;
154.         if (val < rval)
155.             r = r.left;
156.         else if (val > rval)
157.             r = r.right;
158.         else
159.         {
160.             found = true;
161.             break;
162.         }
163.         found = search(r, val);
164.     }
165.     return found;
166. }
167.
168. public void inorder()
169. {
170.     inorder(root);
171. }
172.
173. private void inorder(SBBST r)
174. {
175.     if (r != null)
176.     {
177.         inorder(r.left);
178.         System.out.print(r.data + " ");
179.         inorder(r.right);

```

```

180.     }
181. }
182.
183. public void preorder()
184. {
185.
186.     preorder(root);
187. }
188.
189. private void preorder(SBBST r)
190. {
191.     if (r != null)
192.     {
193.         System.out.print(r.data + " ");
194.         preorder(r.left);
195.         preorder(r.right);
196.     }
197. }
198.
199. public void postorder()
200. {
201.     postorder(root);
202. }
203.
204. private void postorder(SBBST r)
205. {
206.     if (r != null)
207.     {
208.         postorder(r.left);
209.         postorder(r.right);
210.         System.out.print(r.data + " ");
211.     }
212. }
213. }
214.
215. public class Left_Rotation_BST
216. {
217.     public static void main(String[] args)
218.     {
219.         Scanner scan = new Scanner(System.in);
220.
221.         SelfBalancingBinarySearchTree sbbst = new SelfBalancingBinarySearchTree();
222.         System.out.println("Self Balancing Tree\n");
223.
224.         System.out.println("Inset first 10 Elements");
225.         int N = 10;
226.         for (int i = 0; i < N; i++)
227.         {
228.             sbbst.insert(scan.nextInt());
229.
230.             System.out.println("\nPre-order :");
231.             sbbst.preorder();
232.             System.out.println("\nIn-order :");
233.             sbbst.inorder();
234.             System.out.println("\nPost-order :");
235.             sbbst.postorder();
236.
237.             System.out.println();
238.         }
239.         scan.close();
240.     }
241. }

```

Output:

advertisement

```
$ javac Left_Rotation_BST.java  
$ java Left_Rotation_BST
```

Self Balancing Tree

Inset first 10 Elements

10

Pre-order :

10

In-order :

10

Post-order :

10

9

Pre-order :

10 9

In-order :

9 10

Post-order :

9 10

8

Left Rotation Performed

Pre-order :

9 8 10

In-order :

8 9 10

Post-order :

8 10 9

7

Pre-order :

9 8 7 10

In-order :

7 8 9 10

Post-order :

7 8 10 9

6

Left Rotation Performed

Pre-order :

9 7 6 8 10

In-order :

6 7 8 9 10

Post-order :

6 8 7 10 9

5

Left Rotation Performed

Pre-order :

7 6 5 9 8 10

In-order :

5 6 7 8 9 10

Post-order :

5 6 8 10 9 7

4

Left Rotation Performed

Pre-order :

7 5 4 6 9 8 10

In-order :

4 5 6 7 8 9 10

Post-order :

4 6 5 8 10 9 7

3

Pre-order :
7 5 4 3 6 9 8 10

In-order :
3 4 5 6 7 8 9 10
Post-order :
3 4 6 5 8 10 9 7

2

Left Rotation Performed

Pre-order :
7 5 3 2 4 6 9 8 10

In-order :
2 3 4 5 6 7 8 9 10
Post-order :
2 4 3 6 5 8 10 9 7

1

Left Rotation Performed

Pre-order :
7 3 2 1 5 4 6 9 8 10

In-order :
1 2 3 4 5 6 7 8 9 10
Post-order :
1 2 4 6 5 3 8 10 9 7

265.Java Program to Print the Kind of Rotation the AVL Tree is Undergoing When you Add an Element or Delete an Element

[« Prev](#)

[Next »](#)

This is a Java Program to implement Self Balancing Binary Search Tree. A self-balancing (or height-balanced) binary search tree is any node-based binary search tree that automatically keeps its height (maximal number of levels below the root) small in the face of arbitrary item insertions and deletions. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets. The implementation of self balancing binary search tree is similar to that of a AVL Tree data structure.

Here is the source code of the Java Program to Print the Kind of Rotation the AVL Tree is Undergoing When you Add an Element or Delete an Element. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement self balancing binary search trees and indicate when rotation is performed
2. import java.util.Scanner;
3.
4. class SBBST
5. {
6.     SBBST left, right;
7.     int data;
8.     int height;
9.
10.    public SBBST()
11.    {
12.        left = null;
13.        right = null;
14.        data = 0;
15.        height = 0;
16.    }
17.
18.    public SBBST(int n)
19.    {
20.
21.        left = null;
22.        right = null;
23.        data = n;
24.        height = 0;
25.    }
26.
27.
28.class SelfBalancingBinarySearchTree
29.
30.    private SBBST root;
31.
32.    public SelfBalancingBinarySearchTree()
33.    {
34.        root = null;
35.    }
36.
37.    public boolean isEmpty()
38.    {
39.        return root == null;
40.    }
41.
42.    public void clear()
43.    {
```

```

44.     root = null;
45. }
46.
47. public void insert(int data)
48. {
49.     root = insert(data, root);
50. }
51.
52. private int height(SBBST t)
53. {
54.
55.     return t == null ? -1 : t.height;
56. }
57.
58. private int max(int lhs, int rhs)
59. {
60.     return lhs > rhs ? lhs : rhs;
61. }
62.
63. private SBBST insert(int x, SBBST t)
64. {
65.     if (t == null)
66.         t = new SBBST(x);
67.     else if (x < t.data)
68.     {
69.         t.left = insert(x, t.left);
70.         if (height(t.left) - height(t.right) == 2)
71.             if (x < t.left.data)
72.                 t = rotateWithLeftChild(t);
73.             else
74.                 t = doubleWithLeftChild(t);
75.     } else if (x > t.data)
76.     {
77.         t.right = insert(x, t.right);
78.         if (height(t.right) - height(t.left) == 2)
79.             if (x > t.right.data)
80.                 t = rotateWithRightChild(t);
81.             else
82.                 t = doubleWithRightChild(t);
83.     } else
84.     ;
85.     t.height = max(height(t.left), height(t.right)) + 1;
86.     return t;
87. }
88.
89. private SBBST rotateWithLeftChild(SBBST k2)
90. {
91.     System.out.println("Left Rotation Performed");
92.     SBBST k1 = k2.left;
93.     k2.left = k1.right;
94.     k1.right = k2;
95.     k2.height = max(height(k2.left), height(k2.right)) + 1;
96.     k1.height = max(height(k1.left), k2.height) + 1;
97.     return k1;
98. }
99.
100. private SBBST rotateWithRightChild(SBBST k1)
101. {
102.     System.out.println("Right Rotation Performed");
103.     SBBST k2 = k1.right;
104.     k1.right = k2.left;
105.     k2.left = k1;
106.     k1.height = max(height(k1.left), height(k1.right)) + 1;
107.     k2.height = max(height(k2.right), k1.height) + 1;
108.     return k2;
109. }

```

```
110.
111. private SBBST doubleWithLeftChild(SBBST k3)
112. {
113.     System.out.println("Left Rotation Performed");
114.     k3.left = rotateWithRightChild(k3.left);
115.     return rotateWithLeftChild(k3);
116. }
117.
118. private SBBST doubleWithRightChild(SBBST k1)
119. {
120.     System.out.println("Right Rotation Performed");
121.     k1.right = rotateWithLeftChild(k1.right);
122.     return rotateWithRightChild(k1);
123. }
124.
125. public int countNodes()
126. {
127.     return countNodes(root);
128. }
129.
130. private int countNodes(SBBST r)
131. {
132.     if (r == null)
133.         return 0;
134.     else
135.     {
136.         int l = 1;
137.         l += countNodes(r.left);
138.         l += countNodes(r.right);
139.         return l;
140.     }
141. }
142.
143. public boolean search(int val)
144. {
145.     return search(root, val);
146. }
147.
148. private boolean search(SBBST r, int val)
149. {
150.     boolean found = false;
151.     while ((r != null) && !found)
152.     {
153.         int rval = r.data;
154.         if (val < rval)
155.             r = r.left;
156.         else if (val > rval)
157.             r = r.right;
158.         else
159.         {
160.             found = true;
161.             break;
162.         }
163.         found = search(r, val);
164.     }
165.     return found;
166. }
167.
168. public void inorder()
169. {
170.     inorder(root);
171. }
172.
173. private void inorder(SBBST r)
174. {
175.     if (r != null)
```

```

176.     {
177.         inorder(r.left);
178.         System.out.print(r.data + " ");
179.         inorder(r.right);
180.     }
181. }
182.
183. public void preorder()
184. {
185.
186.     preorder(root);
187. }
188.
189. private void preorder(SBBST r)
190. {
191.     if (r != null)
192.     {
193.         System.out.print(r.data + " ");
194.         preorder(r.left);
195.         preorder(r.right);
196.     }
197. }
198.
199. public void postorder()
200. {
201.     postorder(root);
202. }
203.
204. private void postorder(SBBST r)
205. {
206.     if (r != null)
207.     {
208.         postorder(r.left);
209.         postorder(r.right);
210.         System.out.print(r.data + " ");
211.     }
212. }
213. }
214.
215. public class Rotation_BST
216. {
217.     public static void main(String[] args)
218.     {
219.         Scanner scan = new Scanner(System.in);
220.
221.         SelfBalancingBinarySearchTree sbbst = new SelfBalancingBinarySearchTree();
222.         System.out.println("Self Balancing Tree\n");
223.
224.         System.out.println("Inset first 10 Elements");
225.         int N = 10;
226.         for (int i = 0; i < N; i++)
227.         {
228.             sbbst.insert(scan.nextInt());
229.
230.             System.out.println("\nPre-order :");
231.             sbbst.preorder();
232.             System.out.println("\nIn-order :");
233.             sbbst.inorder();
234.             System.out.println("\nPost-order :");
235.             sbbst.postorder();
236.
237.             System.out.println();
238.         }
239.         scan.close();
240.     }
241. }

```

Output:

advertisement

```
$ javac Rotation_BST.java  
$ java Rotation_BST
```

Self Balancing Tree

Inset first 10 Elements

1

Pre-order :

1

In-order :

1

Post-order :

1

2

Pre-order :

1 2

In-order :

1 2

Post-order :

2 1

3

Right Rotation Performed

Pre-order :

2 1 3

In-order :

1 2 3

Post-order :

1 3 2

6

Pre-order :

2 1 3 6

In-order :

1 2 3 6

Post-order :

1 6 3 2

5

Right Rotation Performed

Left Rotation Performed

Right Rotation Performed

Pre-order :

2 1 5 3 6

In-order :

1 2 3 5 6

Post-order :

1 3 6 5 2

4

Right Rotation Performed

Left Rotation Performed

Right Rotation Performed

Pre-order :

3 2 1 5 4 6

In-order :

1 2 3 4 5 6

Post-order :

1 2 4 6 5 3

9

Pre-order :
3 2 1 5 4 6 9
In-order :
1 2 3 4 5 6 9
Post-order :
1 2 4 9 6 5 3
8
Right Rotation Performed
Left Rotation Performed
Right Rotation Performed

Pre-order :
3 2 1 5 4 8 6 9
In-order :
1 2 3 4 5 6 8 9
Post-order :
1 2 4 6 9 8 5 3
7
Right Rotation Performed
Left Rotation Performed
Right Rotation Performed

Pre-order :
3 2 1 6 5 4 8 7 9
In-order :
1 2 3 4 5 6 7 8 9
Post-order :
1 2 4 5 7 9 8 6 3

266.Java Program to Check if a Given Binary Tree is an AVL Tree or Not

[« Prev](#)

[Next »](#)

This is a Java Program to implement a binary tree and check whether it is AVL Tree or not. An AVL tree is a self-balancing binary search tree. It was the first such data structure to be invented. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. Insertions and deletions may require the tree to be rebalanced by one or more tree rotations. A tree is AVL if and only if it is Binary Search Tree and is Balanced.

Here is the source code of the Java Program to Check if a Given Binary Tree is an AVL Tree or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether a tree is AVL tree or not
2. class BSTAVLTreeNode
3. {
4.     int value;
5.     BSTAVLTreeNode Left;
6.     BSTAVLTreeNode Right;
7.
8.     BSTAVLTreeNode(int k)
9.     {
10.         value = k;
11.     }
12. }
13.
14.class BST_AVL
15.
16. public static boolean isBST(BSTAVLTreeNode node)
17. {
18.     return (isBSTUtil(node, 0, 100));
19. }
20.
21. public static boolean isBSTUtil(BSTAVLTreeNode node, int min, int max)
22. {
23.
24.     if (node == null)
25.         return true;
26.
27.     if (node.value < min || node.value > max)
28.         return false;
29.
30.     return (isBSTUtil(node.Left, min, node.value - 1) && isBSTUtil(
31.             node.Right, node.value + 1, max));
32. }
33.
34. public static boolean isBalanced(BSTAVLTreeNode root)
35. {
36.     int lh; /*for height of left subtree */
37.     int rh; /*for height of right subtree */
38.
39.     if (root == null)
40.         return true;
41.
42.     lh = height(root.Left);
43.     rh = height(root.Right);
44.
45.     if (Math.abs(lh - rh) <= 1 && isBalanced(root.Left)
```

```

46.         && isBalanced(root.Right))
47.     return true;
48.
49.    return false;
50. }
51.
52. public static int max(int a, int b)
53. {
54.     return (a >= b) ? a : b;
55. }
56.
57. public static int height(BSTAVLTreeNode node)
58. {
59.     if (node == null)
60.         return 0;
61.
62.     return 1 + max(height(node.Left), height(node.Right));
63. }
64.
65. public static void main(String args[])
66. {
67.     BSTAVLTreeNode t1 = new BSTAVLTreeNode(1);
68.     t1.Left = new BSTAVLTreeNode(2);
69.     t1.Right = new BSTAVLTreeNode(3);
70.     t1.Right.Left = new BSTAVLTreeNode(4);
71.     t1.Right.Right = new BSTAVLTreeNode(5);
72.
73.     BSTAVLTreeNode t2 = new BSTAVLTreeNode(15);
74.     t2.Left = new BSTAVLTreeNode(5);
75.     t2.Right = new BSTAVLTreeNode(20);
76.     t2.Right.Left = new BSTAVLTreeNode(18);
77.     t2.Right.Right = new BSTAVLTreeNode(23);
78.     t2.Left.Left = new BSTAVLTreeNode(4);
79.     t2.Left.Right = new BSTAVLTreeNode(6);
80.
81.     if (isBST(t1) && isBalanced(t1))
82.         System.out.println("Tree t1 is AVL tree");
83.     else
84.         System.out.println("Tree t1 is not AVL tree");
85.
86.     if (isBST(t2) && isBalanced(t2))
87.         System.out.println("Tree t1 is AVL tree");
88.     else
89.         System.out.println("Tree t1 is not AVL tree");
90. }
91. }
```

Output:

advertisement

```
$ javac BST_AVL.java
$ java BST_AVL
```

Tree t1 is not AVL tree
 Tree t1 is AVL tree

267.Java Program to Implement Slicker Algorithm that avoids Triangulation to Find Area of a Polygon

[« Prev](#)

[Next »](#)

This is a Java Program to find the area of a polygon using slicker method. The algorithm assumes the usual mathematical convention that positive y points upwards. In computer systems where positive y is downwards (most of them) the easiest thing to do is list the vertices counter-clockwise using the “positive y down” coordinates. The two effects then cancel out to produce a positive area.

Here is the source code of the Java Program to Implement Slicker Algorithm that avoids Triangulation to Find Area of a Polygon. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the area of polygon using Slicker algorithm
2. import java.util.*;
3.
4. class Area_polygon_Slicker
5. {
6.     static final int MAXPOLY = 200;
7.     static final double EPSILON = 0.000001;
8.
9.     static class Point
10.    {
11.        double x, y;
12.    }
13.
14.     static class Polygon
15.    {
16.        Point p[] = new Point[MAXPOLY];
17.        int n;
18.
19.        Polygon()
20.        {
21.            for (int i = 0; i < MAXPOLY; i++)
22.                p[i] = new Point();
23.        }
24.    }
25.
26.     static double area(Polygon p)
27.    {
28.        double total = 0;
29.        for (int i = 0; i < p.n; i++)
30.        {
31.            int j = (i + 1) % p.n;
32.            total += (p.p[i].x * p.p[j].y - (p.p[j].x * p.p[i].y));
33.        }
34.        return total / 2;
35.    }
36.
37.     static public void main(String[] args)
38.    {
39.        Polygon p = new Polygon();
40.        Scanner sc = new Scanner(System.in);
41.        System.out.println("Enter the number of points in Polygon: ");
42.        p.n = sc.nextInt();
43.        System.out.println("Enter the coordinates of each point: <x> <y>");
44.        for (int i = 0; i < p.n; i++)
45.        {
46.            p.p[i].x = sc.nextDouble();
```

```
47.     p.p[i].y = sc.nextDouble();
48. }
49.
50. double area = area(p);
51. if (area > 0)
52.     System.out.print("The Area of Polygon with " + p.n
53.                     + " points using Slicker Algorithm is : " + area);
54. else
55.     System.out.print("The Area of Polygon with " + p.n
56.                     + " points using Slicker Algorithm is : " + (area * -1));
57. sc.close();
58. }
59. }
```

Output:

advertisement

```
$ javac Area_polygon_Slicker.java
$ java Area_polygon_Slicker
```

Enter the number of points in Polygon:

4

Enter the coordinates of each point: <x> <y>

```
1 1
1 6
6 6
6 1
```

The Area of Polygon with 4 points using Slicker Algorithm is : 25.0

Enter the number of points in Polygon:

5

Enter the coordinates of each point: <x> <y>

```
1 2
4 5
9 8
3 2
1 5
```

The Area of Polygon with 5points using Slicker Algorithm is : 6.0

268.Java Program to Compute the Area of a Triangle Using Determinants

[« Prev](#)

[Next »](#)

This is a Java Program to find the area of a triangle using determinant method. Formula for the area of a triangle using determinants

$$\text{Area} = \pm \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

The plus/minus in this case is meant to take whichever sign is needed so the answer is positive (non-negative). Do not say the area is both positive and negative.

advertisement

Here is the source code of the Java Program to Compute the Area of a Triangle Using Determinants. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the area of triangle using method of determinants
2. import java.util.Random;
3.
4. public class Area_Triangle_Determinants
5. {
6.     public static double determinant(double A[][], int N)
7.     {
8.         double det = 0;
9.         if (N == 1)
10.        {
11.            det = A[0][0];
12.        } else if (N == 2)
13.        {
14.            det = A[0][0] * A[1][1] - A[1][0] * A[0][1];
15.        } else
16.        {
17.            det = 0;
18.            for (int j1 = 0; j1 < N; j1++)
19.            {
20.                double[][] m = new double[N - 1][];
21.                for (int k = 0; k < (N - 1); k++)
22.                {
23.                    m[k] = new double[N - 1];
24.                }
25.                for (int i = 1; i < N; i++)
26.                {
27.                    int j2 = 0;
28.                    for (int j = 0; j < N; j++)
29.                    {
30.                        if (j == j1)
31.                            continue;
32.                        m[i - 1][j2] = A[i][j];
33.                        j2++;
34.                    }
35.                }
36.                det += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
37.                * determinant(m, N - 1);
38.            }
39.        }
40.        return det;
41.    }
```

```

42.
43. public static void main(String args[])
44. {
45.     Random random = new Random();
46.     int x1, x2, x3, y1, y2, y3;
47.     x1 = random.nextInt(10);
48.     x2 = random.nextInt(10);
49.     x3 = random.nextInt(10);
50.     y1 = random.nextInt(10);
51.     y2 = random.nextInt(10);
52.     y3 = random.nextInt(10);
53.
54.     double[][] mat = new double[3][3];
55.     mat[0][0] = x1;
56.     mat[0][1] = y1;
57.     mat[0][2] = 1;
58.     mat[1][0] = x2;
59.     mat[1][1] = y2;
60.     mat[1][2] = 1;
61.     mat[2][0] = x3;
62.     mat[2][1] = y3;
63.     mat[2][2] = 1;
64.
65.     System.out
66.         .println("The matrix formed by the coordinates of the triangle is:");
67.     for (int i = 0; i < 3; i++)
68.     {
69.         for (int j = 0; j < 3; j++)
70.             System.out.print(mat[i][j] + " ");
71.         System.out.println();
72.     }
73.
74.     double det = determinant(mat, 3) * 0.5;
75.     if (det < 0)
76.         System.out.println("The Area of the triangle formed by (" + x1
77.                         + "," + y1 + "), (" + x2 + "," + y2 + "), (" + x3 + ","
78.                         + y3 + ") = " + (det * -1));
79.     else
80.         System.out.println("The Area of the triangle formed by (" + x1
81.                         + "," + y1 + "), (" + x2 + "," + y2 + "), (" + x3 + ","
82.                         + y3 + ") = " + det);
83.    }
84.}

```

Output:

advertisement

```
$ javac Area_Triangle_Determinants.java
$ java Area_Triangle_Determinants
```

The matrix formed by the coordinates of the triangle is:

```
3.0 4.0 1.0
6.0 4.0 1.0
3.0 9.0 1.0
```

The Area of the triangle formed by (3,4), (6,4), (3,9) = 7.5

269.Java Program to Check if a Given Set of Three Points Lie on a Single Line or Not

[« Prev](#)

[Next »](#)

This is a Java Program to check whether three points are collinear or not. We do this by taking two points make an equation of the line passing through those two points and check whether third points lies on it. In geometry, collinearity is a property of a set of points, specifically, the property of lying on a single line.

Here is the source code of the Java Program to Check if a Given Set of Three Points Lie on a Single Line or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether three points are collinear or not
2. import java.util.Scanner;
3.
4. public class Collinear_Points
5. {
6.     public static void main(String args[])
7.     {
8.         System.out.println("Enter the points : <x>,<y>");
9.         Scanner scan = new Scanner(System.in);
10.        int x, y, x1, x2, y1, y2;
11.        x = scan.nextInt();
12.        y = scan.nextInt();
13.        x1 = scan.nextInt();
14.        x2 = scan.nextInt();
15.        y1 = scan.nextInt();
16.        y2 = scan.nextInt();
17.
18.        /*
19.         * System.out.println("The Equation of the line is : (" + (y2 - y1) +
20.         * ")x + (" + (x1 - x2) + ")y + (" + (x2 * y1 - x1 * y2) + ") = 0");
21.         */
22.
23.        int s = (y2 - y1) * x + (x1 - x2) * y + (x2 * y1 - x1 * y2);
24.        if (s < 0)
25.            System.out.println("The points are NOT collinear");
26.        else if (s > 0)
27.            System.out.println("The points are NOT collinear");
28.        else
29.            System.out.println("The points are collinear");
30.        scan.close();
31.    }
32.}
```

Output:

advertisement

```
$ javac Collinear_Points.java
$ java Collinear_Points
```

Enter the points : <x>,<y>

```
3 2
1 3
1 5
```

The points are NOT collinear

Enter the points : <x>,<y>

```
1 1
1 5
1 9
```

The points are collinear

270.Java Program to Compute Cross Product of Two Vectors

[« Prev](#)

[Next »](#)

This is a Java Program to compute cross product of two vectors. In mathematics, the cross product or vector product is a binary operation on two vectors in three-dimensional space. It results in a vector that is perpendicular to both and therefore normal to the plane containing them.

Here is the source code of the Java Program to Compute Cross Product of Two Vectors. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the cross product of two vectors
2. import java.util.Random;
3.
4. public class Cross_Product
5. {
6.     public static void main(String args[])
7.     {
8.         Random random = new Random();
9.         int u1, u2, u3, v1, v2, v3;
10.        u1 = random.nextInt(10);
11.        u2 = random.nextInt(10);
12.        u3 = random.nextInt(10);
13.        v1 = random.nextInt(10);
14.        v2 = random.nextInt(10);
15.        v3 = random.nextInt(10);
16.
17.        int uv_i, uv_j, uv_k;
18.        uv_i = u2 * v3 - v2 * u3;
19.        uv_j = v1 * u3 - u1 * v3;
20.        uv_k = u1 * v2 - v1 * u2;
21.
22.        System.out.println("The cross product of the 2 vectors \n u = " + u1
23.                           + "i + " + u2 + "j + " + u3 + "k and \n v = " + v1 + "i +
24.                           + v2 + "j + " + v3 + "k \n ");
25.        System.out.println("u X v : " + uv_i + "i + " + uv_j + "j + " + uv_k + "k ");
26.    }
27. }
```

Output:

advertisement

```
$ javac Cross_Product.java
$ java Cross_Product
```

The cross product of the 2 vectors

$u = 3i + 8j + 9k$ and

$v = 3i + 8j + 9k$

```
u X v : -2i +48j+ -42k
```

271.Java Program for Douglas-Peucker Algorithm Implementation

[« Prev](#)

[Next »](#)

This is a Java Program to implement Douglas-Peucker Algorithm. The Ramer–Douglas–Peucker algorithm (RDP) is an algorithm for reducing the number of points in a curve that is approximated by a series of points. This algorithm is also known under the names Douglas–Peucker algorithm, iterative end-point fit algorithm and split-and-merge algorithm. The purpose of the algorithm is, given a curve composed of line segments, to find a similar curve with fewer points. The algorithm defines ‘dissimilar’ based on the maximum distance between the original curve and the simplified curve. The simplified curve consists of a subset of the points that defined the original curve.

Here is the source code of the Java Program for Douglas-Peucker Algorithm Implementation. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to filter out points using Douglas Peucker Algorithm
2. import static java.lang.Math.abs;
3. import static java.lang.Math.pow;
4. import static java.lang.Math.sqrt;
5.
6. import java.util.Random;
7.
8. class RamerDouglasPeuckerFilter
9. {
10.
11.     private double epsilon;
12.
13.     public RamerDouglasPeuckerFilter(double epsilon)
14.     {
15.         if (epsilon <= 0)
16.         {
17.             throw new IllegalArgumentException("Epsilon must be > 0");
18.         }
19.         this.epsilon = epsilon;
20.     }
21.
22.     public double[] filter(double[] data)
23.     {
24.         return ramerDouglasPeuckerFunction(data, 0, data.length - 1);
25.     }
26.
27.     public double getEpsilon()
28.     {
29.         return epsilon;
30.     }
31.
32.     protected double[] ramerDouglasPeuckerFunction(double[] points,
33.         int startIndex, int endIndex)
34.     {
35.         double dmax = 0;
36.         int idx = 0;
37.         double a = endIndex - startIndex;
38.         double b = points[endIndex] - points[startIndex];
39.         double c = -(b * startIndex - a * points[startIndex]);
40.         double norm = sqrt(pow(a, 2) + pow(b, 2));
41.         for (int i = startIndex + 1; i < endIndex; i++)
42.         {
43.             double distance = abs(b * i - a * points[i] + c) / norm;
44.             if (distance > dmax)
45.             {
```

```

46.         idx = i;
47.         dmax = distance;
48.     }
49. }
50. if (dmax >= epsilon)
51. {
52.     double[] recursiveResult1 = ramerDouglasPeuckerFunction(points,
53.         startIndex, idx);
54.     double[] recursiveResult2 = ramerDouglasPeuckerFunction(points,
55.         idx, endIndex);
56.     double[] result = new double[(recursiveResult1.length - 1)
57.         + recursiveResult2.length];
58.     System.arraycopy(recursiveResult1, 0, result, 0,
59.         recursiveResult1.length - 1);
60.     System.arraycopy(recursiveResult2, 0, result,
61.         recursiveResult1.length - 1, recursiveResult2.length);
62.     return result;
63. } else
64. {
65.     return new double[] { points[startIndex], points[endIndex] };
66. }
67. }
68.
69. public void setEpsilon(double epsilon)
70. {
71.     if (epsilon <= 0)
72.     {
73.         throw new IllegalArgumentException("Epsilon must be > 0");
74.     }
75.     this.epsilon = epsilon;
76. }
77.
78.
79.
80. public class Douglas_Peucker_Algorithm
81. {
82.     public static void main(String args[])
83.     {
84.         Random random = new Random();
85.         double[] points = new double[20];
86.         double[] fpoints;
87.         for (int i = 0; i < points.length; i++)
88.             points[i] = random.nextInt(10);
89.         RamerDouglasPeuckerFilter rdpf = new RamerDouglasPeuckerFilter();
90.         fpoints = rdpf.filter(points);
91.
92.         System.out.println("Orginal points");
93.         for (int i = 0; i < points.length; i++)
94.             System.out.print(points[i] + " ");
95.
96.         System.out.println("\nFiltered points");
97.         for (int i = 0; i < fpoints.length; i++)
98.             System.out.print(fpoints[i] + " ");
99.     }
100. }

```

Output:

advertisement

```
$ javac Douglas_Peucker_Algorithm.java
$ java Douglas_Peucker_Algorithm
```

```
Orginal points
5.0 0.0 8.0 7.0 2.0 9.0 4.0 4.0 0.0 7.0 4.0 1.0 9.0 6.0 8.0 9.0 6.0 6.0 9.0 6.0
Filtered points
5.0 0.0 8.0 2.0 9.0 0.0 7.0 1.0 9.0 6.0 9.0 6.0 9.0 6.0
```

272. Java Program to Find Nearest Neighbor for Dynamic Data Set

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and find the nearest neighbor for dynamic input set. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Find Nearest Neighbor for Dynamic Data Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find nearest neighbor for dynamic data set
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KDN
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KDN Parent;
14.    KDN Left;
15.    KDN Right;
16.
17.    public KDN(double[] x0, int axis0)
18.    {
19.        x = new double[2];
20.        axis = axis0;
21.        for (int k = 0; k < 2; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KDN FindParent(double[] x0)
30.    {
31.        KDN parent = null;
32.        KDN next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KDN Insert(double[] p)
47.    {
48.        x = new double[2];
```

```

49.     KDN parent = FindParent(p);
50.     if (equal(p, parent.x, 2) == true)
51.         return null;
52.
53.     KDN newNode = new KDN(p, parent.axis + 1 < 2 ? parent.axis + 1 : 0);
54.     newNode.Parent = parent;
55.
56.     if (p[parent.axis] > parent.x[parent.axis])
57.     {
58.         parent.Right = newNode;
59.         newNode.orientation = true; //
60.     } else
61.     {
62.         parent.Left = newNode;
63.         newNode.orientation = false; //
64.     }
65.
66.     return newNode;
67. }
68.
69. boolean equal(double[] x1, double[] x2, int dim)
70. {
71.     for (int k = 0; k < dim; k++)
72.     {
73.         if (x1[k] != x2[k])
74.             return false;
75.     }
76.
77.     return true;
78. }
79.
80. double distance2(double[] x1, double[] x2, int dim)
81. {
82.     double S = 0;
83.     for (int k = 0; k < dim; k++)
84.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
85.     return S;
86. }
87.}
88.
89.class KDTreeDynamic
90.{
91.    KDN Root;
92.
93.    int TimeStart, TimeFinish;
94.    int CounterFreq;
95.
96.    double d_min;
97.    KDN nearest_neighbour;
98.
99.    int KD_id;
100.
101.   int nList;
102.
103.   KDN CheckedNodes[];
104.   int checked_nodes;
105.   KDN List[];
106.
107.   double x_min[], x_max[];
108.   boolean max_boundary[], min_boundary[];
109.   int n_boundary;
110.
111.  public KDTreeDynamic(int i)
112.  {
113.      Root = null;
114.      KD_id = 1;

```

```

115.     nList = 0;
116.     List = new KDN[i];
117.     CheckedNodes = new KDN[i];
118.     max_boundary = new boolean[2];
119.     min_boundary = new boolean[2];
120.     x_min = new double[2];
121.     x_max = new double[2];
122. }
123.
124. public boolean add(double[] x)
125. {
126.     if (nList >= 2000000 - 1)
127.         return false; // can't add more points
128.
129.     if (Root == null)
130.     {
131.         Root = new KDN(x, 0);
132.         Root.id = KD_id++;
133.         List[nList++] = Root;
134.     } else
135.     {
136.         KDN pNode;
137.         if ((pNode = Root.Insert(x)) != null)
138.         {
139.             pNode.id = KD_id++;
140.             List[nList++] = pNode;
141.         }
142.     }
143.
144.     return true;
145. }
146.
147. public KDN find_nearest(double[] x)
148. {
149.     if (Root == null)
150.         return null;
151.
152.     checked_nodes = 0;
153.     KDN parent = Root.FindParent(x);
154.     nearest_neighbour = parent;
155.     d_min = Root.distance2(x, parent.x, 2);
156. ;
157.
158.     if (parent.equal(x, parent.x, 2) == true)
159.         return nearest_neighbour;
160.
161.     search_parent(parent, x);
162.     uncheck();
163.
164.     return nearest_neighbour;
165. }
166.
167. public void check_subtree(KDN node, double[] x)
168. {
169.     if ((node == null) || node.checked)
170.         return;
171.
172.     CheckedNodes[checked_nodes++] = node;
173.     node.checked = true;
174.     set_bounding_cube(node, x);
175.
176.     int dim = node.axis;
177.     double d = node.x[dim] - x[dim];
178.
179.     if (d * d > d_min)
180.     {

```

```

181.     if (node.x[dim] > x[dim])
182.         check_subtree(node.Left, x);
183.     else
184.         check_subtree(node.Right, x);
185. } else
186. {
187.     check_subtree(node.Left, x);
188.     check_subtree(node.Right, x);
189. }
190. }
191.
192. public void set_bounding_cube(KDN node, double[] x)
193. {
194.     if (node == null)
195.         return;
196.     int d = 0;
197.     double dx;
198.     for (int k = 0; k < 2; k++)
199.     {
200.         dx = node.x[k] - x[k];
201.         if (dx > 0)
202.         {
203.             dx *= dx;
204.             if (!max_boundary[k])
205.             {
206.                 if (dx > x_max[k])
207.                     x_max[k] = dx;
208.                 if (x_max[k] > d_min)
209.                 {
210.                     max_boundary[k] = true;
211.                     n_boundary++;
212.                 }
213.             }
214.         } else
215.         {
216.             dx *= dx;
217.             if (!min_boundary[k])
218.             {
219.                 if (dx > x_min[k])
220.                     x_min[k] = dx;
221.                 if (x_min[k] > d_min)
222.                 {
223.                     min_boundary[k] = true;
224.                     n_boundary++;
225.                 }
226.             }
227.         }
228.         d += dx;
229.         if (d > d_min)
230.             return;
231.     }
232. }
233.
234. if (d < d_min)
235. {
236.     d_min = d;
237.     nearest_neighbour = node;
238. }
239. }
240.
241. public KDN search_parent(KDN parent, double[] x)
242. {
243.     for (int k = 0; k < 2; k++)
244.     {
245.         x_min[k] = x_max[k] = 0;
246.         max_boundary[k] = min_boundary[k] = false; //

```

```

247.     }
248.     n_boundary = 0;
249.
250.     KDN search_root = parent;
251.     while (parent != null && (n_boundary != 2 * 2))
252.     {
253.         check_subtree(parent, x);
254.         search_root = parent;
255.         parent = parent.Parent;
256.     }
257.
258.     return search_root;
259. }
260.
261. public void uncheck()
262. {
263.     for (int n = 0; n < checked_nodes; n++)
264.         CheckedNodes[n].checked = false;
265. }
266.
267. }
268.
269. public class Dynamic_Nearest
270. {
271.
272.     public static void main(String args[]) throws IOException
273.     {
274.         int numpoints = 10;
275.         Scanner sc = new Scanner(System.in);
276.         KDTreeDynamic kdt = new KDTreeDynamic(numpoints);
277.         double x[] = new double[2];
278.
279.         System.out.println("Enter the first 10 data set : <x> <y>");
280.         for (int i = 0; i < numpoints; i++)
281.         {
282.             x[0] = sc.nextDouble();
283.             x[1] = sc.nextDouble();
284.             kdt.add(x);
285.         }
286.
287.         System.out.println("Enter the co-ordinates of the point: <x> <y>");
288.
289.         double sx = sc.nextDouble();
290.         double sy = sc.nextDouble();
291.
292.         double s[] = { sx, sy };
293.         KDN kdn = kdt.find_nearest(s);
294.         System.out.println("The nearest neighbor for the static data set is: ");
295.         System.out.println("(" + kdn.x[0] + ", " + kdn.x[1] + ")");
296.         sc.close();
297.     }
298. }

```

Output:

advertisement

```
$ javac Dynamic_Nearest.java
$ java Dynamic_Nearest
```

Enter the first 10 data set :

```
1.2 3.3
2.3 3.4
4.5 5.6
6.7 7.8
8.9 9.0
10.1 11.3
```

15.6 19.4
20.5 25.4
52.8 65.3
62.6 56.3

Enter the co-ordinates of the point: <x> <y>
60 34.2

The nearest neighbor for the static data set is:
(62.6 , 56.3)

273. Java Program to Find Nearest Neighbor for Static Data Set

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and find the nearest neighbor for static input set. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Find Nearest Neighbor for Static Data Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the nearest neighbor for the static data set
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KDNodes
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KDNodes Parent;
14.    KDNodes Left;
15.    KDNodes Right;
16.
17.    public KDNodes(double[] x0, int axis0)
18.    {
19.        x = new double[2];
20.        axis = axis0;
21.        for (int k = 0; k < 2; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KDNodes FindParent(double[] x0)
30.    {
31.        KDNodes parent = null;
32.        KDNodes next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KDNodes Insert(double[] p)
47.    {
48.        x = new double[2];
```

```

49.     KDNodes parent = FindParent(p);
50.     if (equal(p, parent.x, 2) == true)
51.         return null;
52.
53.     KDNodes newNode = new KDNodes(p, parent.axis + 1 < 2 ? parent.axis + 1
54.                                     : 0);
55.     newNode.Parent = parent;
56.
57.     if (p[parent.axis] > parent.x[parent.axis])
58.     {
59.         parent.Right = newNode;
60.         newNode.orientation = true; //
61.     } else
62.     {
63.         parent.Left = newNode;
64.         newNode.orientation = false; //
65.     }
66.
67.     return newNode;
68. }
69.
70. boolean equal(double[] x1, double[] x2, int dim)
71. {
72.     for (int k = 0; k < dim; k++)
73.     {
74.         if (x1[k] != x2[k])
75.             return false;
76.     }
77.
78.     return true;
79. }
80.
81. double distance2(double[] x1, double[] x2, int dim)
82. {
83.     double S = 0;
84.     for (int k = 0; k < dim; k++)
85.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
86.     return S;
87. }
88.
89.
90.class KDTreeStatic
91.
92.     KDNodes Root;
93.
94.     int TimeStart, TimeFinish;
95.     int CounterFreq;
96.
97.     double d_min;
98.     KDNodes nearest_neighbour;
99.
100.    int KD_id;
101.
102.    int nList;
103.
104.    KDNodes CheckedNodes[];
105.    int checked_nodes;
106.    KDNodes List[];
107.
108.    double x_min[], x_max[];
109.    boolean max_boundary[], min_boundary[];
110.    int n_boundary;
111.
112.    public KDTreeStatic(int i)
113.    {
114.        Root = null;

```

```

115.     KD_id = 1;
116.     nList = 0;
117.     List = new KDNodes[i];
118.     CheckedNodes = new KDNodes[i];
119.     max_boundary = new boolean[2];
120.     min_boundary = new boolean[2];
121.     x_min = new double[2];
122.     x_max = new double[2];
123. }
124.
125. public boolean add(double[] x)
126. {
127.     if (nList >= 2000000 - 1)
128.         return false; // can't add more points
129.
130.     if (Root == null)
131.     {
132.         Root = new KDNodes(x, 0);
133.         Root.id = KD_id++;
134.         List[nList++] = Root;
135.     } else
136.     {
137.         KDNodes pNode;
138.         if ((pNode = Root.Insert(x)) != null)
139.         {
140.             pNode.id = KD_id++;
141.             List[nList++] = pNode;
142.         }
143.     }
144.
145.     return true;
146. }
147.
148. public KDNodes find_nearest(double[] x)
149. {
150.     if (Root == null)
151.         return null;
152.
153.     checked_nodes = 0;
154.     KDNodes parent = Root.FindParent(x);
155.     nearest_neighbour = parent;
156.     d_min = Root.distance2(x, parent.x, 2);
157. ;
158.
159.     if (parent.equal(x, parent.x, 2) == true)
160.         return nearest_neighbour;
161.
162.     search_parent(parent, x);
163.     uncheck();
164.
165.     return nearest_neighbour;
166. }
167.
168. public void check_subtree(KDNodes node, double[] x)
169. {
170.     if ((node == null) || node.checked)
171.         return;
172.
173.     CheckedNodes[checked_nodes++] = node;
174.     node.checked = true;
175.     set_bounding_cube(node, x);
176.
177.     int dim = node.axis;
178.     double d = node.x[dim] - x[dim];
179.
180.     if (d * d > d_min)

```

```

181.     {
182.         if (node.x[dim] > x[dim])
183.             check_subtree(node.Left, x);
184.         else
185.             check_subtree(node.Right, x);
186.     } else
187.     {
188.         check_subtree(node.Left, x);
189.         check_subtree(node.Right, x);
190.     }
191. }
192.
193. public void set_bounding_cube(KDNodes node, double[] x)
194. {
195.     if (node == null)
196.         return;
197.     int d = 0;
198.     double dx;
199.     for (int k = 0; k < 2; k++)
200.     {
201.         dx = node.x[k] - x[k];
202.         if (dx > 0)
203.         {
204.             dx *= dx;
205.             if (!max_boundary[k])
206.             {
207.                 if (dx > x_max[k])
208.                     x_max[k] = dx;
209.                 if (x_max[k] > d_min)
210.                 {
211.                     max_boundary[k] = true;
212.                     n_boundary++;
213.                 }
214.             }
215.         } else
216.         {
217.             dx *= dx;
218.             if (!min_boundary[k])
219.             {
220.                 if (dx > x_min[k])
221.                     x_min[k] = dx;
222.                 if (x_min[k] > d_min)
223.                 {
224.                     min_boundary[k] = true;
225.                     n_boundary++;
226.                 }
227.             }
228.         }
229.         d += dx;
230.         if (d > d_min)
231.             return;
232.     }
233. }
234.
235. if (d < d_min)
236. {
237.     d_min = d;
238.     nearest_neighbour = node;
239. }
240. }
241.
242. public KDNodes search_parent(KDNodes parent, double[] x)
243. {
244.     for (int k = 0; k < 2; k++)
245.     {
246.         x_min[k] = x_max[k] = 0;

```

```

247.     max_boundary[k] = min_boundary[k] = false; //
248. }
249. n_boundary = 0;
250.
251. KDNodes search_root = parent;
252. while (parent != null && (n_boundary != 2 * 2))
253. {
254.     check_subtree(parent, x);
255.     search_root = parent;
256.     parent = parent.Parent;
257. }
258.
259. return search_root;
260. }
261.
262. public void uncheck()
263. {
264.     for (int n = 0; n < checked_nodes; n++)
265.         CheckedNodes[n].checked = false;
266. }
267.
268. }
269.
270. public class Static_Nearest
271. {
272.
273.     public static void main(String args[]) throws IOException
274.     {
275.         int numpoints = 7;
276.         Scanner sc = new Scanner(System.in);
277.         KDTreeStatic kdt = new KDTreeStatic(numpoints);
278.         double x[] = new double[2];
279.
280.         x[0] = 2.1;
281.         x[1] = 4.3;
282.         kdt.add(x);
283.
284.         x[0] = 3.3;
285.         x[1] = 1.5;
286.         kdt.add(x);
287.
288.         x[0] = 4.7;
289.         x[1] = 11.1;
290.         kdt.add(x);
291.
292.         x[0] = 5.0;
293.         x[1] = 12.3;
294.         kdt.add(x);
295.
296.         x[0] = 5.1;
297.         x[1] = 1.2;
298.         kdt.add(x);
299.
300.         x[0] = 12.1;
301.         x[1] = 18.2;
302.         kdt.add(x);
303.
304.         x[0] = 20.5;
305.         x[1] = 17.9;
306.         kdt.add(x);
307.
308.         System.out.println("Enter the co-ordinates of the point: <x> <y>");
309.
310.         double sx = sc.nextDouble();
311.         double sy = sc.nextDouble();
312.

```

```
313.     double s[] = { sx, sy };
314.     KDNodes kdn = kdt.find_nearest(s);
315.     System.out.println("The nearest neighbor for the static data set is: ");
316.     System.out.println("(" + kdn.x[0] + ", " + kdn.x[1] + ")");
317.     sc.close();
318. }
319. }
```

Output:

advertisement

```
$ javac Static_Nearest.java
$ java Static_Nearest
```

Enter the co-ordinates of the point: <x> <y>

9 9

The nearest neighbor for the static data set is:

(4.7 , 11.1)

Enter the co-ordinates of the point: <x> <y>

5 20

The nearest neighbor for the static data set is:

(12.1 , 18.2)

274. Java Program to Perform Insertion in a 2 Dimension K-D Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and insert the input set and print the various traversals. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Perform Insertion in a 2 Dimension K-D Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to insert an element in a 2D KD Tree
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KD2DNode
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KD2DNode Parent;
14.    KD2DNode Left;
15.    KD2DNode Right;
16.
17.    public KD2DNode(double[] x0, int axis0)
18.    {
19.        x = new double[2];
20.        axis = axis0;
21.        for (int k = 0; k < 2; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KD2DNode FindParent(double[] x0)
30.    {
31.        KD2DNode parent = null;
32.        KD2DNode next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KD2DNode Insert(double[] p)
47.    {
48.        x = new double[2];
```

```

49.     KD2DNode parent = FindParent(p);
50.     if (equal(p, parent.x, 2) == true)
51.         return null;
52.
53.     KD2DNode newNode = new KD2DNode(p,
54.         parent.axis + 1 < 2 ? parent.axis + 1 : 0);
55.     newNode.Parent = parent;
56.
57.     if (p[parent.axis] > parent.x[parent.axis])
58.     {
59.         parent.Right = newNode;
60.         newNode.orientation = true; //
61.     } else
62.     {
63.         parent.Left = newNode;
64.         newNode.orientation = false; //
65.     }
66.
67.     return newNode;
68. }
69.
70. boolean equal(double[] x1, double[] x2, int dim)
71. {
72.     for (int k = 0; k < dim; k++)
73.     {
74.         if (x1[k] != x2[k])
75.             return false;
76.     }
77.
78.     return true;
79. }
80.
81. double distance2(double[] x1, double[] x2, int dim)
82. {
83.     double S = 0;
84.     for (int k = 0; k < dim; k++)
85.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
86.     return S;
87. }
88.
89.
90.class KD2DTree
91.
92.     KD2DNode Root;
93.
94.     int TimeStart, TimeFinish;
95.     int CounterFreq;
96.
97.     double d_min;
98.     KD2DNode nearest_neighbour;
99.
100.    int KD_id;
101.
102.    int nList;
103.
104.    KD2DNode CheckedNodes[];
105.    int checked_nodes;
106.    KD2DNode List[];
107.
108.    double x_min[], x_max[];
109.    boolean max_boundary[], min_boundary[];
110.    int n_boundary;
111.
112.    public KD2DTree(int i)
113.    {
114.        Root = null;

```

```

115.     KD_id = 1;
116.     nList = 0;
117.     List = new KD2DNode[i];
118.     CheckedNodes = new KD2DNode[i];
119.     max_boundary = new boolean[2];
120.     min_boundary = new boolean[2];
121.     x_min = new double[2];
122.     x_max = new double[2];
123. }
124.
125. public boolean add(double[] x)
126. {
127.     if (nList >= 2000000 - 1)
128.         return false; // can't add more points
129.
130.     if (Root == null)
131.     {
132.         Root = new KD2DNode(x, 0);
133.         Root.id = KD_id++;
134.         List[nList++] = Root;
135.     } else
136.     {
137.         KD2DNode pNode;
138.         if ((pNode = Root.Insert(x)) != null)
139.         {
140.             pNode.id = KD_id++;
141.             List[nList++] = pNode;
142.         }
143.     }
144.
145.     return true;
146. }
147.
148. public KD2DNode find_nearest(double[] x)
149. {
150.     if (Root == null)
151.         return null;
152.
153.     checked_nodes = 0;
154.     KD2DNode parent = Root.FindParent(x);
155.     nearest_neighbour = parent;
156.     d_min = Root.distance2(x, parent.x, 2);
157. ;
158.
159.     if (parent.equal(x, parent.x, 2) == true)
160.         return nearest_neighbour;
161.
162.     search_parent(parent, x);
163.     uncheck();
164.
165.     return nearest_neighbour;
166. }
167.
168. public void check_subtree(KD2DNode node, double[] x)
169. {
170.     if ((node == null) || node.checked)
171.         return;
172.
173.     CheckedNodes[checked_nodes++] = node;
174.     node.checked = true;
175.     set_bounding_cube(node, x);
176.
177.     int dim = node.axis;
178.     double d = node.x[dim] - x[dim];
179.
180.     if (d * d > d_min)

```

```

181.     {
182.         if (node.x[dim] > x[dim])
183.             check_subtree(node.Left, x);
184.         else
185.             check_subtree(node.Right, x);
186.     } else
187.     {
188.         check_subtree(node.Left, x);
189.         check_subtree(node.Right, x);
190.     }
191. }
192.
193. public void set_bounding_cube(KD2DNode node, double[] x)
194. {
195.     if (node == null)
196.         return;
197.     int d = 0;
198.     double dx;
199.     for (int k = 0; k < 2; k++)
200.     {
201.         dx = node.x[k] - x[k];
202.         if (dx > 0)
203.         {
204.             dx *= dx;
205.             if (!max_boundary[k])
206.             {
207.                 if (dx > x_max[k])
208.                     x_max[k] = dx;
209.                 if (x_max[k] > d_min)
210.                 {
211.                     max_boundary[k] = true;
212.                     n_boundary++;
213.                 }
214.             }
215.         } else
216.         {
217.             dx *= dx;
218.             if (!min_boundary[k])
219.             {
220.                 if (dx > x_min[k])
221.                     x_min[k] = dx;
222.                 if (x_min[k] > d_min)
223.                 {
224.                     min_boundary[k] = true;
225.                     n_boundary++;
226.                 }
227.             }
228.         }
229.         d += dx;
230.         if (d > d_min)
231.             return;
232.     }
233. }
234.
235. if (d < d_min)
236. {
237.     d_min = d;
238.     nearest_neighbour = node;
239. }
240. }
241.
242. public KD2DNode search_parent(KD2DNode parent, double[] x)
243. {
244.     for (int k = 0; k < 2; k++)
245.     {
246.         x_min[k] = x_max[k] = 0;

```

```

247.     max_boundary[k] = min_boundary[k] = false; //  

248. }
249. n_boundary = 0;  

250.  

251. KD2DNode search_root = parent;  

252. while (parent != null && (n_boundary != 2 * 2))  

253. {
254.     check_subtree(parent, x);  

255.     search_root = parent;  

256.     parent = parent.Parent;  

257. }
258.  

259. return search_root;  

260. }  

261.  

262. public void uncheck()  

263. {
264.     for (int n = 0; n < checked_nodes; n++)  

265.         CheckedNodes[n].checked = false;  

266. }
267.  

268. public void inorder()  

269. {
270.     inorder(Root);  

271. }
272.  

273. private void inorder(KD2DNode root)  

274. {
275.     if (root != null)
276.     {
277.         inorder(root.Left);
278.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ")");
279.         inorder(root.Right);
280.     }
281. }
282.  

283. public void preorder()  

284. {
285.     preorder(Root);
286. }
287.  

288. private void preorder(KD2DNode root)  

289. {
290.     if (root != null)
291.     {
292.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");
293.         inorder(root.Left);
294.         inorder(root.Right);
295.     }
296. }
297.  

298. public void postorder()  

299. {
300.     postorder(Root);
301. }
302.  

303. private void postorder(KD2DNode root)  

304. {
305.     if (root != null)
306.     {
307.         inorder(root.Left);
308.         inorder(root.Right);
309.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ")");
310.     }
311. }
312. }

```

```

313.
314.public class KDTree_TwoD_Data
315. {
316.   public static void main(String args[]) throws IOException
317.   {
318.     int numpoints = 5;
319.     Scanner sc = new Scanner(System.in);
320.     KD2DTree kdt = new KD2DTree(numpoints);
321.     double x[] = new double[2];
322.     System.out.println("Enter the first 5 data set : <x> <y>");
323.     for (int i = 0; i < numpoints; i++)
324.     {
325.       x[0] = sc.nextDouble();
326.       x[1] = sc.nextDouble();
327.       kdt.add(x);
328.     }
329.
330.     System.out.println("Inorder of 2D Kd tree: ");
331.     kdt.inorder();
332.
333.     System.out.println("\nPreorder of 2D Kd tree: ");
334.     kdt.preorder();
335.
336.     System.out.println("\nPostorder of 2D Kd tree: ");
337.     kdt.postorder();
338.     sc.close();
339.   }
340. }
```

Output:

advertisement

```
$ javac KD2D_Insertion.java
$ java KD2D_Insertion
```

Enter the first 10 data set : <x> <y>

```
0 0
2 3
3 4
4 5
5 6
```

Inorder of 2D Kd tree:

```
(0.0, 0.0) (2.0, 3.0) (3.0, 4.0) (4.0, 5.0) (5.0, 6.0)
```

Preorder of 2D Kd tree:

```
(0.0, 0.0) (2.0, 3.0) (3.0, 4.0) (4.0, 5.0) (5.0, 6.0)
```

Postorder of 2D Kd tree:

```
(2.0, 3.0) (3.0, 4.0) (4.0, 5.0) (5.0, 6.0) (0.0, 0.0)
```

275. Java Program to Construct K-D Tree for 2 Dimensional Data (assume static data)

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and print the various traversals. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Construct K-D Tree for 2 Dimensional Data (assume static data). The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to construct a KD tree for two dimensional static data
2. import java.io.IOException;
3.
4. class KD2DNode
5. {
6.     int axis;
7.     double[] x;
8.     int id;
9.     boolean checked;
10.    boolean orientation;
11.
12.    KD2DNode Parent;
13.    KD2DNode Left;
14.    KD2DNode Right;
15.
16.    public KD2DNode(double[] x0, int axis0)
17.    {
18.        x = new double[2];
19.        axis = axis0;
20.        for (int k = 0; k < 2; k++)
21.            x[k] = x0[k];
22.
23.        Left = Right = Parent = null;
24.        checked = false;
25.        id = 0;
26.    }
27.
28.    public KD2DNode FindParent(double[] x0)
29.    {
30.        KD2DNode parent = null;
31.        KD2DNode next = this;
32.        int split;
33.        while (next != null)
34.        {
35.            split = next.axis;
36.            parent = next;
37.            if (x0[split] > next.x[split])
38.                next = next.Right;
39.            else
40.                next = next.Left;
41.        }
42.        return parent;
43.    }
44.
45.    public KD2DNode Insert(double[] p)
46.    {
47.        x = new double[2];
```

```

48.     KD2DNode parent = FindParent(p);
49.     if (equal(p, parent.x, 2) == true)
50.         return null;
51.
52.     KD2DNode newNode = new KD2DNode(p,
53.         parent.axis + 1 < 2 ? parent.axis + 1 : 0);
54.     newNode.Parent = parent;
55.
56.     if (p[parent.axis] > parent.x[parent.axis])
57.     {
58.         parent.Right = newNode;
59.         newNode.orientation = true; //
60.     } else
61.     {
62.         parent.Left = newNode;
63.         newNode.orientation = false; //
64.     }
65.
66.     return newNode;
67. }
68.
69. boolean equal(double[] x1, double[] x2, int dim)
70. {
71.     for (int k = 0; k < dim; k++)
72.     {
73.         if (x1[k] != x2[k])
74.             return false;
75.     }
76.
77.     return true;
78. }
79.
80. double distance2(double[] x1, double[] x2, int dim)
81. {
82.     double S = 0;
83.     for (int k = 0; k < dim; k++)
84.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
85.     return S;
86. }
87. }
88.
89.class KD2DTree
90.{
91.    KD2DNode Root;
92.
93.    int TimeStart, TimeFinish;
94.    int CounterFreq;
95.
96.    double d_min;
97.    KD2DNode nearest_neighbour;
98.
99.    int KD_id;
100.
101.   int nList;
102.
103.   KD2DNode CheckedNodes[];
104.   int checked_nodes;
105.   KD2DNode List[];
106.
107.   double x_min[], x_max[];
108.   boolean max_boundary[], min_boundary[];
109.   int n_boundary;
110.
111.  public KD2DTree(int i)
112.  {
113.      Root = null;

```

```

114.     KD_id = 1;
115.     nList = 0;
116.     List = new KD2DNode[i];
117.     CheckedNodes = new KD2DNode[i];
118.     max_boundary = new boolean[2];
119.     min_boundary = new boolean[2];
120.     x_min = new double[2];
121.     x_max = new double[2];
122. }
123.
124. public boolean add(double[] x)
125. {
126.     if (nList >= 2000000 - 1)
127.         return false; // can't add more points
128.
129.     if (Root == null)
130.     {
131.         Root = new KD2DNode(x, 0);
132.         Root.id = KD_id++;
133.         List[nList++] = Root;
134.     } else
135.     {
136.         KD2DNode pNode;
137.         if ((pNode = Root.Insert(x)) != null)
138.         {
139.             pNode.id = KD_id++;
140.             List[nList++] = pNode;
141.         }
142.     }
143.
144.     return true;
145. }
146.
147. public KD2DNode find_nearest(double[] x)
148. {
149.     if (Root == null)
150.         return null;
151.
152.     checked_nodes = 0;
153.     KD2DNode parent = Root.FindParent(x);
154.     nearest_neighbour = parent;
155.     d_min = Root.distance2(x, parent.x, 2);
156. ;
157.
158.     if (parent.equal(x, parent.x, 2) == true)
159.         return nearest_neighbour;
160.
161.     search_parent(parent, x);
162.     uncheck();
163.
164.     return nearest_neighbour;
165. }
166.
167. public void check_subtree(KD2DNode node, double[] x)
168. {
169.     if ((node == null) || node.checked)
170.         return;
171.
172.     CheckedNodes[checked_nodes++] = node;
173.     node.checked = true;
174.     set_bounding_cube(node, x);
175.
176.     int dim = node.axis;
177.     double d = node.x[dim] - x[dim];
178.
179.     if (d * d > d_min)

```

```

180.     {
181.         if (node.x[dim] > x[dim])
182.             check_subtree(node.Left, x);
183.         else
184.             check_subtree(node.Right, x);
185.     } else
186.     {
187.         check_subtree(node.Left, x);
188.         check_subtree(node.Right, x);
189.     }
190. }
191.
192. public void set_bounding_cube(KD2DNode node, double[] x)
193. {
194.     if (node == null)
195.         return;
196.     int d = 0;
197.     double dx;
198.     for (int k = 0; k < 2; k++)
199.     {
200.         dx = node.x[k] - x[k];
201.         if (dx > 0)
202.         {
203.             dx *= dx;
204.             if (!max_boundary[k])
205.             {
206.                 if (dx > x_max[k])
207.                     x_max[k] = dx;
208.                 if (x_max[k] > d_min)
209.                 {
210.                     max_boundary[k] = true;
211.                     n_boundary++;
212.                 }
213.             }
214.         } else
215.         {
216.             dx *= dx;
217.             if (!min_boundary[k])
218.             {
219.                 if (dx > x_min[k])
220.                     x_min[k] = dx;
221.                 if (x_min[k] > d_min)
222.                 {
223.                     min_boundary[k] = true;
224.                     n_boundary++;
225.                 }
226.             }
227.         }
228.         d += dx;
229.         if (d > d_min)
230.             return;
231.     }
232. }
233.
234. if (d < d_min)
235. {
236.     d_min = d;
237.     nearest_neighbour = node;
238. }
239. }
240.
241. public KD2DNode search_parent(KD2DNode parent, double[] x)
242. {
243.     for (int k = 0; k < 2; k++)
244.     {
245.         x_min[k] = x_max[k] = 0;

```

```

246.     max_boundary[k] = min_boundary[k] = false; //  

247. }
248. n_boundary = 0;  

249.  

250. KD2DNode search_root = parent;  

251. while (parent != null && (n_boundary != 2 * 2))  

252. {
253.     check_subtree(parent, x);
254.     search_root = parent;
255.     parent = parent.Parent;
256. }
257.  

258. return search_root;
259. }
260.  

261. public void uncheck()
262. {
263.     for (int n = 0; n < checked_nodes; n++)
264.         CheckedNodes[n].checked = false;
265. }
266.  

267. public void inorder()
268. {
269.     inorder(Root);
270. }
271.  

272. private void inorder(KD2DNode root)
273. {
274.     if (root != null)
275.     {
276.         inorder(root.Left);
277.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ")");
278.         inorder(root.Right);
279.     }
280. }
281.  

282. public void preorder()
283. {
284.     preorder(Root);
285. }
286.  

287. private void preorder(KD2DNode root)
288. {
289.     if (root != null)
290.     {
291.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");
292.         inorder(root.Left);
293.         inorder(root.Right);
294.     }
295. }
296.  

297. public void postorder()
298. {
299.     postorder(Root);
300. }
301.  

302. private void postorder(KD2DNode root)
303. {
304.     if (root != null)
305.     {
306.         inorder(root.Left);
307.         inorder(root.Right);
308.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ")");
309.     }
310. }
311. }

```

```

312.
313.public class KDTree_TwoD_Data
314. {
315.   public static void main(String args[]) throws IOException
316.   {
317.     int numpoints = 5;
318.
319.     KD2DTree kdt = new KD2DTree(numpoints);
320.     double x[] = new double[2];
321.
322.     x[0] = 0.0;
323.     x[1] = 0.0;
324.     kdt.add(x);
325.
326.     x[0] = 3.3;
327.     x[1] = 1.5;
328.     kdt.add(x);
329.
330.     x[0] = 4.7;
331.     x[1] = 11.1;
332.     kdt.add(x);
333.
334.     x[0] = 5.0;
335.     x[1] = 12.3;
336.     kdt.add(x);
337.
338.     x[0] = 5.1;
339.     x[1] = 1.2;
340.     kdt.add(x);
341.
342.     System.out.println("Inorder of 2D Kd tree: ");
343.     kdt.inorder();
344.
345.     System.out.println("\nPreorder of 2D Kd tree: ");
346.     kdt.preorder();
347.
348.     System.out.println("\nPostorder of 2D Kd tree: ");
349.     kdt.postorder();
350.   }
351. }
```

Output:

advertisement

```
$ javac KDTree_TwoD_Data.java
$ java KDTree_TwoD_Data
```

```
Inorder of 2D Kd tree:
(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)
Preorder of 2D Kd tree:
(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)
Postorder of 2D Kd tree:
(5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3) (0.0, 0.0)
```

276. Java Program to Perform Partial Key Search in a K-D Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and perform partial search(Searching a node with either of the coordinate). In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Perform Partial Key Search in a K-D Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform partial search in 2D KD Tree
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KD2DNode
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KD2DNode Parent;
14.    KD2DNode Left;
15.    KD2DNode Right;
16.
17.    public KD2DNode(double[] x0, int axis0)
18.    {
19.        x = new double[2];
20.        axis = axis0;
21.        for (int k = 0; k < 2; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KD2DNode FindParent(double[] x0)
30.    {
31.        KD2DNode parent = null;
32.        KD2DNode next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KD2DNode Insert(double[] p)
47.    {
48.        x = new double[2];
```

```

49.     KD2DNode parent = FindParent(p);
50.     if (equal(p, parent.x, 2) == true)
51.         return null;
52.
53.     KD2DNode newNode = new KD2DNode(p,
54.         parent.axis + 1 < 2 ? parent.axis + 1 : 0);
55.     newNode.Parent = parent;
56.
57.     if (p[parent.axis] > parent.x[parent.axis])
58.     {
59.         parent.Right = newNode;
60.         newNode.orientation = true; //
61.     } else
62.     {
63.         parent.Left = newNode;
64.         newNode.orientation = false; //
65.     }
66.
67.     return newNode;
68. }
69.
70. boolean equal(double[] x1, double[] x2, int dim)
71. {
72.     for (int k = 0; k < dim; k++)
73.     {
74.         if (x1[k] != x2[k])
75.             return false;
76.     }
77.
78.     return true;
79. }
80.
81. double distance2(double[] x1, double[] x2, int dim)
82. {
83.     double S = 0;
84.     for (int k = 0; k < dim; k++)
85.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
86.     return S;
87. }
88.
89.
90.class KD2DTree
91.
92.     KD2DNode Root;
93.
94.     int TimeStart, TimeFinish;
95.     int CounterFreq;
96.
97.     double d_min;
98.     KD2DNode nearest_neighbour;
99.
100.    int KD_id;
101.
102.    int nList;
103.
104.    KD2DNode CheckedNodes[];
105.    int checked_nodes;
106.    KD2DNode List[];
107.
108.    double x_min[], x_max[];
109.    boolean max_boundary[], min_boundary[];
110.    int n_boundary;
111.
112.    public KD2DTree(int i)
113.    {
114.        Root = null;

```

```

115.     KD_id = 1;
116.     nList = 0;
117.     List = new KD2DNode[i];
118.     CheckedNodes = new KD2DNode[i];
119.     max_boundary = new boolean[2];
120.     min_boundary = new boolean[2];
121.     x_min = new double[2];
122.     x_max = new double[2];
123. }
124.
125. public boolean add(double[] x)
126. {
127.     if (nList >= 2000000 - 1)
128.         return false; // can't add more points
129.
130.     if (Root == null)
131.     {
132.         Root = new KD2DNode(x, 0);
133.         Root.id = KD_id++;
134.         List[nList++] = Root;
135.     } else
136.     {
137.         KD2DNode pNode;
138.         if ((pNode = Root.Insert(x)) != null)
139.         {
140.             pNode.id = KD_id++;
141.             List[nList++] = pNode;
142.         }
143.     }
144.
145.     return true;
146. }
147.
148. public KD2DNode find_nearest(double[] x)
149. {
150.     if (Root == null)
151.         return null;
152.
153.     checked_nodes = 0;
154.     KD2DNode parent = Root.FindParent(x);
155.     nearest_neighbour = parent;
156.     d_min = Root.distance2(x, parent.x, 2);
157. ;
158.
159.     if (parent.equal(x, parent.x, 2) == true)
160.         return nearest_neighbour;
161.
162.     search_parent(parent, x);
163.     uncheck();
164.
165.     return nearest_neighbour;
166. }
167.
168. public void check_subtree(KD2DNode node, double[] x)
169. {
170.     if ((node == null) || node.checked)
171.         return;
172.
173.     CheckedNodes[checked_nodes++] = node;
174.     node.checked = true;
175.     set_bounding_cube(node, x);
176.
177.     int dim = node.axis;
178.     double d = node.x[dim] - x[dim];
179.
180.     if (d * d > d_min)

```

```

181.     {
182.         if (node.x[dim] > x[dim])
183.             check_subtree(node.Left, x);
184.         else
185.             check_subtree(node.Right, x);
186.     } else
187.     {
188.         check_subtree(node.Left, x);
189.         check_subtree(node.Right, x);
190.     }
191. }
192.
193. public void set_bounding_cube(KD2DNode node, double[] x)
194. {
195.     if (node == null)
196.         return;
197.     int d = 0;
198.     double dx;
199.     for (int k = 0; k < 2; k++)
200.     {
201.         dx = node.x[k] - x[k];
202.         if (dx > 0)
203.         {
204.             dx *= dx;
205.             if (!max_boundary[k])
206.             {
207.                 if (dx > x_max[k])
208.                     x_max[k] = dx;
209.                 if (x_max[k] > d_min)
210.                 {
211.                     max_boundary[k] = true;
212.                     n_boundary++;
213.                 }
214.             }
215.         } else
216.         {
217.             dx *= dx;
218.             if (!min_boundary[k])
219.             {
220.                 if (dx > x_min[k])
221.                     x_min[k] = dx;
222.                 if (x_min[k] > d_min)
223.                 {
224.                     min_boundary[k] = true;
225.                     n_boundary++;
226.                 }
227.             }
228.         }
229.         d += dx;
230.         if (d > d_min)
231.             return;
232.     }
233. }
234.
235. if (d < d_min)
236. {
237.     d_min = d;
238.     nearest_neighbour = node;
239. }
240. }
241.
242. public KD2DNode search_parent(KD2DNode parent, double[] x)
243. {
244.     for (int k = 0; k < 2; k++)
245.     {
246.         x_min[k] = x_max[k] = 0;

```

```

247.     max_boundary[k] = min_boundary[k] = false; //  

248. }  

249. n_boundary = 0;  

250.  

251. KD2DNode search_root = parent;  

252. while (parent != null && (n_boundary != 2 * 2))  

253. {  

254.     check_subtree(parent, x);  

255.     search_root = parent;  

256.     parent = parent.Parent;  

257. }  

258.  

259. return search_root;  

260.}  

261.  

262. public void uncheck()  

263. {  

264.     for (int n = 0; n < checked_nodes; n++)  

265.         CheckedNodes[n].checked = false;  

266. }  

267.  

268. public void inorder()  

269. {  

270.     inorder(Root);  

271. }  

272.  

273. private void inorder(KD2DNode root)  

274. {  

275.     if (root != null)  

276.     {  

277.         inorder(root.Left);  

278.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");  

279.         inorder(root.Right);  

280.     }  

281. }  

282.  

283. public void preorder()  

284. {  

285.     preorder(Root);  

286. }  

287.  

288. private void preorder(KD2DNode root)  

289. {  

290.     if (root != null)  

291.     {  

292.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");  

293.         inorder(root.Left);  

294.         inorder(root.Right);  

295.     }  

296. }  

297.  

298. public void postorder()  

299. {  

300.     postorder(Root);  

301. }  

302.  

303. private void postorder(KD2DNode root)  

304. {  

305.     if (root != null)  

306.     {  

307.         inorder(root.Left);  

308.         inorder(root.Right);  

309.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");  

310.     }  

311. }  

312.

```

```

313. public void search(double p)
314. {
315.     search(Root, p);
316. }
317.
318. private void search(KD2DNode root, double p)
319. {
320.     if (root != null)
321.     {
322.         search(root.Left, p);
323.         if (p == root.x[0] || p == root.x[1])
324.             System.out.print("True (" + root.x[0] + ", " + root.x[1]
325.                             + ")");
326.         search(root.Right, p);
327.     }
328. }
329. }
330.
331. public class KD2D_Partial_Search
332. {
333.     public static void main(String args[]) throws IOException
334.     {
335.         int numpoints = 5;
336.         Scanner sc = new Scanner(System.in);
337.         KD2DTree kdt = new KD2DTree(numpoints);
338.         double x[] = new double[2];
339.
340.         x[0] = 0.0;
341.         x[1] = 0.0;
342.         kdt.add(x);
343.
344.         x[0] = 3.3;
345.         x[1] = 1.5;
346.         kdt.add(x);
347.
348.         x[0] = 4.7;
349.         x[1] = 11.1;
350.         kdt.add(x);
351.
352.         x[0] = 5.0;
353.         x[1] = 12.3;
354.         kdt.add(x);
355.
356.         x[0] = 5.1;
357.         x[1] = 1.2;
358.         kdt.add(x);
359.
360.         System.out.println("Enter the any one of the co-ordinates of the point: <x>/<y>");
361.         double q = sc.nextDouble();
362.
363.
364.         kdt.search(q);
365.
366.         System.out.println("\nInorder of 2D Kd tree: ");
367.         kdt.inorder();
368.
369.         System.out.println("\nPreorder of 2D Kd tree: ");
370.         kdt.preorder();
371.
372.         System.out.println("\npostorder of 2D Kd tree: ");
373.         kdt.postorder();
374.         sc.close();
375.     }
376. }
```

Output:

```
$ javac KD2D_Partial_Search.java  
$ java KD2D_Partial_Search
```

Partial Key Search

Enter the any one of the co-ordinates of the point: <x>/<y>

5

True (5.0, 12.3)

Inorder of 2D Kd tree:

(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)

Preorder of 2D Kd tree:

(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)

postorder of 2D Kd tree:

(5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3) (0.0, 0.0)

277. Java Program to Find the Nearest Neighbor Using K-D Tree Search

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and find nearest neighbor. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Find the Nearest Neighbor Using K-D Tree Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find nearest neighbor using KD Tree implementation
2. import java.io.BufferedReader;
3. import java.io.FileReader;
4. import java.io.IOException;
5. import java.io.InputStreamReader;
6.
7. class KDNode
8. {
9.     int axis;
10.    double[] x;
11.    int id;
12.    boolean checked;
13.    boolean orientation;
14.
15.    KDNode Parent;
16.    KDNode Left;
17.    KDNode Right;
18.
19.    public KDNode(double[] x0, int axis0)
20.    {
21.        x = new double[2];
22.        axis = axis0;
23.        for (int k = 0; k < 2; k++)
24.            x[k] = x0[k];
25.
26.        Left = Right = Parent = null;
27.        checked = false;
28.        id = 0;
29.    }
30.
31.    public KDNode FindParent(double[] x0)
32.    {
33.        KDNode parent = null;
34.        KDNode next = this;
35.        int split;
36.        while (next != null)
37.        {
38.            split = next.axis;
39.            parent = next;
40.            if (x0[split] > next.x[split])
41.                next = next.Right;
42.            else
43.                next = next.Left;
44.        }
45.        return parent;
46.    }
47.
48.    public KDNode Insert(double[] p)
49.    {
```

```

50.    //x = new double[2];
51.    KDNode parent = FindParent(p);
52.    if (equal(p, parent.x, 2) == true)
53.        return null;
54.
55.    KDNode newNode = new KDNode(p, parent.axis + 1 < 2 ? parent.axis + 1
56.                                : 0);
57.    newNode.Parent = parent;
58.
59.    if (p[parent.axis] > parent.x[parent.axis])
60.    {
61.        parent.Right = newNode;
62.        newNode.orientation = true; //
63.    } else
64.    {
65.        parent.Left = newNode;
66.        newNode.orientation = false; //
67.    }
68.
69.    return newNode;
70. }
71.
72. boolean equal(double[] x1, double[] x2, int dim)
73. {
74.     for (int k = 0; k < dim; k++)
75.     {
76.         if (x1[k] != x2[k])
77.             return false;
78.     }
79.
80.     return true;
81. }
82.
83. double distance2(double[] x1, double[] x2, int dim)
84. {
85.     double S = 0;
86.     for (int k = 0; k < dim; k++)
87.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
88.     return S;
89. }
90. }
91.
92.class KDTree
93.{
94.    KDNode Root;
95.
96.    int TimeStart, TimeFinish;
97.    int CounterFreq;
98.
99.    double d_min;
100.   KDNode nearest_neighbour;
101.
102.   int KD_id;
103.
104.   int nList;
105.
106.   KDNode CheckedNodes[];
107.   int checked_nodes;
108.   KDNode List[];
109.
110.  double x_min[], x_max[];
111.  boolean max_boundary[], min_boundary[];
112.  int n_boundary;
113.
114.  public KDTree(int i)
115.  {

```

```

116.     Root = null;
117.     KD_id = 1;
118.     nList = 0;
119.     List = new KDNode[i];
120.     CheckedNodes = new KDNode[i];
121.     max_boundary = new boolean[2];
122.     min_boundary = new boolean[2];
123.     x_min = new double[2];
124.     x_max = new double[2];
125. }
126.
127. public boolean add(double[] x)
128. {
129.     if (nList >= 2000000 - 1)
130.         return false; // can't add more points
131.
132.     if (Root == null)
133.     {
134.         Root = new KDNode(x, 0);
135.         Root.id = KD_id++;
136.         List[nList++] = Root;
137.     } else
138.     {
139.         KDNode pNode;
140.         if ((pNode = Root.Insert(x)) != null)
141.         {
142.             pNode.id = KD_id++;
143.             List[nList++] = pNode;
144.         }
145.     }
146.
147.     return true;
148. }
149.
150. public KDNode find_nearest(double[] x)
151. {
152.     if (Root == null)
153.         return null;
154.
155.     checked_nodes = 0;
156.     KDNode parent = Root.FindParent(x);
157.     nearest_neighbour = parent;
158.     d_min = Root.distance2(x, parent.x, 2);
159. ;
160.
161.     if (parent.equal(x, parent.x, 2) == true)
162.         return nearest_neighbour;
163.
164.     search_parent(parent, x);
165.     uncheck();
166.
167.     return nearest_neighbour;
168. }
169.
170. public void check_subtree(KDNode node, double[] x)
171. {
172.     if ((node == null) || node.checked)
173.         return;
174.
175.     CheckedNodes[checked_nodes++] = node;
176.     node.checked = true;
177.     set_bounding_cube(node, x);
178.
179.     int dim = node.axis;
180.     double d = node.x[dim] - x[dim];
181.

```

```

182.     if (d * d > d_min)
183.     {
184.         if (node.x[dim] > x[dim])
185.             check_subtree(node.Left, x);
186.         else
187.             check_subtree(node.Right, x);
188.     } else
189.     {
190.         check_subtree(node.Left, x);
191.         check_subtree(node.Right, x);
192.     }
193. }
194.
195. public void set_bounding_cube(KDNode node, double[] x)
196. {
197.     if (node == null)
198.         return;
199.     int d = 0;
200.     double dx;
201.     for (int k = 0; k < 2; k++)
202.     {
203.         dx = node.x[k] - x[k];
204.         if (dx > 0)
205.         {
206.             dx *= dx;
207.             if (!max_boundary[k])
208.             {
209.                 if (dx > x_max[k])
210.                     x_max[k] = dx;
211.                 if (x_max[k] > d_min)
212.                 {
213.                     max_boundary[k] = true;
214.                     n_boundary++;
215.                 }
216.             }
217.         } else
218.         {
219.             dx *= dx;
220.             if (!min_boundary[k])
221.             {
222.                 if (dx > x_min[k])
223.                     x_min[k] = dx;
224.                 if (x_min[k] > d_min)
225.                 {
226.                     min_boundary[k] = true;
227.                     n_boundary++;
228.                 }
229.             }
230.         }
231.         d += dx;
232.         if (d > d_min)
233.             return;
234.
235.     }
236.
237.     if (d < d_min)
238.     {
239.         d_min = d;
240.         nearest_neighbour = node;
241.     }
242. }
243.
244. public KDNode search_parent(KDNode parent, double[] x)
245. {
246.     for (int k = 0; k < 2; k++)
247.     {

```

```

248.     x_min[k] = x_max[k] = 0;
249.     max_boundary[k] = min_boundary[k] = false; //
250. }
251. n_boundary = 0;
252.
253. KDNode search_root = parent;
254. while (parent != null && (n_boundary != 2 * 2))
255. {
256.     check_subtree(parent, x);
257.     search_root = parent;
258.     parent = parent.Parent;
259. }
260.
261. return search_root;
262. }
263.
264. public void uncheck()
265. {
266.     for (int n = 0; n < checked_nodes; n++)
267.         CheckedNodes[n].checked = false;
268. }
269.
270. }
271.
272. public class KDTNearest
273. {
274.
275.     public static void main(String args[]) throws IOException
276.     {
277.         BufferedReader in = new BufferedReader(new FileReader("input.txt"));
278.         int numpoints = 5;
279.
280.         KDTTree kdt = new KDTTree(numpoints);
281.         double x[] = new double[2];
282.
283.         x[0] = 2.1;
284.         x[1] = 4.3;
285.         kdt.add(x);
286.
287.         x[0] = 3.3;
288.         x[1] = 1.5;
289.         kdt.add(x);
290.
291.         x[0] = 4.7;
292.         x[1] = 11.1;
293.         kdt.add(x);
294.
295.         x[0] = 5.0;
296.         x[1] = 12.3;
297.         kdt.add(x);
298.
299.         x[0] = 5.1;
300.         x[1] = 1.2;
301.         kdt.add(x);
302.
303.         System.out
304.             .println("Enter the co-ordinates of the point: (one after the other)");
305.         InputStreamReader reader = new InputStreamReader(System.in);
306.         BufferedReader br = new BufferedReader(reader);
307.         double sx = Double.parseDouble(br.readLine());
308.         double sy = Double.parseDouble(br.readLine());
309.
310.         double s[] = { sx, sy };
311.         KDNode kdn = kdt.find_nearest(s);
312.         System.out.println("The nearest neighbor is:");
313.         System.out.println("(" + kdn.x[0] + " , " + kdn.x[1] + ")");

```

```
314.     in.close();
315. }
316.}
```

Output:

advertisement

```
$ javac KDTNearest.java
$ java KDTNearest
```

Enter the co-ordinates of the point: (one after the other)

4.3

1.5

The nearest neighbor is:

(5.1 , 1.2)

278. Java Program to Perform Searching in a 2-Dimension K-D Tree

[« Prev](#)

[Next »](#)

This is a Java Program to implement 2D KD Tree and Search an element. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Perform Searching in a 2-Dimension K-D Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to search an element in a 2D KD Tree
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KD2DNode
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KD2DNode Parent;
14.    KD2DNode Left;
15.    KD2DNode Right;
16.
17.    public KD2DNode(double[] x0, int axis0)
18.    {
19.        x = new double[2];
20.        axis = axis0;
21.        for (int k = 0; k < 2; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KD2DNode FindParent(double[] x0)
30.    {
31.        KD2DNode parent = null;
32.        KD2DNode next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KD2DNode Insert(double[] p)
47.    {
48.        x = new double[2];
49.        KD2DNode parent = FindParent(p);
```

```

50.    if (equal(p, parent.x, 2) == true)
51.        return null;
52.
53.    KD2DNode newNode = new KD2DNode(p,
54.        parent.axis + 1 < 2 ? parent.axis + 1 : 0);
55.    newNode.Parent = parent;
56.
57.    if (p[parent.axis] > parent.x[parent.axis])
58.    {
59.        parent.Right = newNode;
60.        newNode.orientation = true; //
61.    } else
62.    {
63.        parent.Left = newNode;
64.        newNode.orientation = false; //
65.    }
66.
67.    return newNode;
68. }
69.
70. boolean equal(double[] x1, double[] x2, int dim)
71. {
72.     for (int k = 0; k < dim; k++)
73.     {
74.         if (x1[k] != x2[k])
75.             return false;
76.     }
77.
78.     return true;
79. }
80.
81. double distance2(double[] x1, double[] x2, int dim)
82. {
83.     double S = 0;
84.     for (int k = 0; k < dim; k++)
85.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
86.     return S;
87. }
88.}
89.
90.class KD2DTree
91.{
92.    KD2DNode Root;
93.
94.    int TimeStart, TimeFinish;
95.    int CounterFreq;
96.
97.    double d_min;
98.    KD2DNode nearest_neighbour;
99.
100.   int KD_id;
101.
102.   int nList;
103.
104.   KD2DNode CheckedNodes[];
105.   int checked_nodes;
106.   KD2DNode List[];
107.
108.   double x_min[], x_max[];
109.   boolean max_boundary[], min_boundary[];
110.   int n_boundary;
111.
112.   public KD2DTree(int i)
113.   {
114.       Root = null;
115.       KD_id = 1;

```

```

116.     nList = 0;
117.     List = new KD2DNode[i];
118.     CheckedNodes = new KD2DNode[i];
119.     max_boundary = new boolean[2];
120.     min_boundary = new boolean[2];
121.     x_min = new double[2];
122.     x_max = new double[2];
123. }
124.
125. public boolean add(double[] x)
126. {
127.     if (nList >= 2000000 - 1)
128.         return false; // can't add more points
129.
130.     if (Root == null)
131.     {
132.         Root = new KD2DNode(x, 0);
133.         Root.id = KD_id++;
134.         List[nList++] = Root;
135.     } else
136.     {
137.         KD2DNode pNode;
138.         if ((pNode = Root.Insert(x)) != null)
139.         {
140.             pNode.id = KD_id++;
141.             List[nList++] = pNode;
142.         }
143.     }
144.
145.     return true;
146. }
147.
148. public KD2DNode find_nearest(double[] x)
149. {
150.     if (Root == null)
151.         return null;
152.
153.     checked_nodes = 0;
154.     KD2DNode parent = Root.FindParent(x);
155.     nearest_neighbour = parent;
156.     d_min = Root.distance2(x, parent.x, 2);
157. ;
158.
159.     if (parent.equal(x, parent.x, 2) == true)
160.         return nearest_neighbour;
161.
162.     search_parent(parent, x);
163.     uncheck();
164.
165.     return nearest_neighbour;
166. }
167.
168. public void check_subtree(KD2DNode node, double[] x)
169. {
170.     if ((node == null) || node.checked)
171.         return;
172.
173.     CheckedNodes[checked_nodes++] = node;
174.     node.checked = true;
175.     set_bounding_cube(node, x);
176.
177.     int dim = node.axis;
178.     double d = node.x[dim] - x[dim];
179.
180.     if (d * d > d_min)
181.     {

```

```

182.     if (node.x[dim] > x[dim])
183.         check_subtree(node.Left, x);
184.     else
185.         check_subtree(node.Right, x);
186. } else
187. {
188.     check_subtree(node.Left, x);
189.     check_subtree(node.Right, x);
190. }
191. }
192.
193. public void set_bounding_cube(KD2DNode node, double[] x)
194. {
195.     if (node == null)
196.         return;
197.     int d = 0;
198.     double dx;
199.     for (int k = 0; k < 2; k++)
200.     {
201.         dx = node.x[k] - x[k];
202.         if (dx > 0)
203.         {
204.             dx *= dx;
205.             if (!max_boundary[k])
206.             {
207.                 if (dx > x_max[k])
208.                     x_max[k] = dx;
209.                 if (x_max[k] > d_min)
210.                 {
211.                     max_boundary[k] = true;
212.                     n_boundary++;
213.                 }
214.             }
215.         } else
216.         {
217.             dx *= dx;
218.             if (!min_boundary[k])
219.             {
220.                 if (dx > x_min[k])
221.                     x_min[k] = dx;
222.                 if (x_min[k] > d_min)
223.                 {
224.                     min_boundary[k] = true;
225.                     n_boundary++;
226.                 }
227.             }
228.         }
229.         d += dx;
230.         if (d > d_min)
231.             return;
232.     }
233. }
234.
235. if (d < d_min)
236. {
237.     d_min = d;
238.     nearest_neighbour = node;
239. }
240. }
241.
242. public KD2DNode search_parent(KD2DNode parent, double[] x)
243. {
244.     for (int k = 0; k < 2; k++)
245.     {
246.         x_min[k] = x_max[k] = 0;
247.         max_boundary[k] = min_boundary[k] = false; //

```

```

248.     }
249.     n_boundary = 0;
250.
251.     KD2DNode search_root = parent;
252.     while (parent != null && (n_boundary != 2 * 2))
253.     {
254.         check_subtree(parent, x);
255.         search_root = parent;
256.         parent = parent.Parent;
257.     }
258.
259.     return search_root;
260. }
261.
262. public void uncheck()
263. {
264.     for (int n = 0; n < checked_nodes; n++)
265.         CheckedNodes[n].checked = false;
266. }
267.
268. public void inorder()
269. {
270.     inorder(Root);
271. }
272.
273. private void inorder(KD2DNode root)
274. {
275.     if (root != null)
276.     {
277.         inorder(root.Left);
278.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");
279.         inorder(root.Right);
280.     }
281. }
282.
283. public void preorder()
284. {
285.     preorder(Root);
286. }
287.
288. private void preorder(KD2DNode root)
289. {
290.     if (root != null)
291.     {
292.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ") ");
293.         inorder(root.Left);
294.         inorder(root.Right);
295.     }
296. }
297.
298. public void postorder()
299. {
300.     postorder(Root);
301. }
302.
303. private void postorder(KD2DNode root)
304. {
305.     if (root != null)
306.     {
307.         inorder(root.Left);
308.         inorder(root.Right);
309.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ")");
310.     }
311. }
312.
313. public void search(double x, double y)

```

```

314. {
315.     search(Root, x, y);
316. }
317.
318. private void search(KD2DNode root, double x, double y)
319. {
320.     if (root != null)
321.     {
322.         search(root.Left, x, y);
323.         if (x == root.x[0] && y == root.x[1])
324.             System.out.print("True (" + root.x[0] + ", " + root.x[1]
325.                             + ")");
326.         search(root.Right, x, y);
327.     }
328. }
329. }
330.
331.public class KD2D_Search
332. {
333.     public static void main(String args[]) throws IOException
334.     {
335.         int numpoints = 5;
336.         Scanner sc = new Scanner(System.in);
337.         KD2DTree kdt = new KD2DTree(numpoints);
338.         double x[] = new double[2];
339.
340.         x[0] = 0.0;
341.         x[1] = 0.0;
342.         kdt.add(x);
343.
344.         x[0] = 3.3;
345.         x[1] = 1.5;
346.         kdt.add(x);
347.
348.         x[0] = 4.7;
349.         x[1] = 11.1;
350.         kdt.add(x);
351.
352.         x[0] = 5.0;
353.         x[1] = 12.3;
354.         kdt.add(x);
355.
356.         x[0] = 5.1;
357.         x[1] = 1.2;
358.         kdt.add(x);
359.
360.         System.out.println("Enter the co-ordinates of the point: <x> <y>");
361.         double x1 = sc.nextDouble();
362.         double y1 = sc.nextDouble();
363.
364.         kdt.search(x1, y1);
365.
366.         System.out.println("\nInorder of 2D Kd tree: ");
367.         kdt.inorder();
368.
369.         System.out.println("\nPreorder of 2D Kd tree: ");
370.         kdt.preorder();
371.
372.         System.out.println("\nPostorder of 2D Kd tree: ");
373.         kdt.postorder();
374.         sc.close();
375.     }
376. }

```

Output:

advertisement

```
$ javac KD2D_Search.java  
$ java KD2D_Search
```

Enter the co-ordinates of the point: <x> <y>

5.1 1.2

True (5.1, 1.2)

Inorder of 2D Kd tree:

(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)

Preorder of 2D Kd tree:

(0.0, 0.0) (5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3)

postorder of 2D Kd tree:

(5.1, 1.2) (3.3, 1.5) (4.7, 11.1) (5.0, 12.3) (0.0, 0.0)

279. Java Program to Find Location of a Point Placed in Three Dimensions Using K-D Trees

[« Prev](#)

[Next »](#)

This is a Java Program to implement 3D KD Tree and Search an element. In computer science, a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees.

Here is the source code of the Java Program to Find Location of a Point Placed in Three Dimensions Using K-D Trees. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the location of point in 3 dimensional KD Tree
2. import java.io.IOException;
3. import java.util.Scanner;
4.
5. class KD3DNode
6. {
7.     int axis;
8.     double[] x;
9.     int id;
10.    boolean checked;
11.    boolean orientation;
12.
13.    KD3DNode Parent;
14.    KD3DNode Left;
15.    KD3DNode Right;
16.
17.    public KD3DNode(double[] x0, int axis0)
18.    {
19.        x = new double[3];
20.        axis = axis0;
21.        for (int k = 0; k < 3; k++)
22.            x[k] = x0[k];
23.
24.        Left = Right = Parent = null;
25.        checked = false;
26.        id = 0;
27.    }
28.
29.    public KD3DNode FindParent(double[] x0)
30.    {
31.        KD3DNode parent = null;
32.        KD3DNode next = this;
33.        int split;
34.        while (next != null)
35.        {
36.            split = next.axis;
37.            parent = next;
38.            if (x0[split] > next.x[split])
39.                next = next.Right;
40.            else
41.                next = next.Left;
42.        }
43.        return parent;
44.    }
45.
46.    public KD3DNode Insert(double[] p)
47.    {
48.        x = new double[3];
```

```

49.     KD3DNode parent = FindParent(p);
50.     if (equal(p, parent.x, 3) == true)
51.         return null;
52.
53.     KD3DNode newNode = new KD3DNode(p,
54.         parent.axis + 1 < 3 ? parent.axis + 1 : 0);
55.     newNode.Parent = parent;
56.
57.     if (p[parent.axis] > parent.x[parent.axis])
58.     {
59.         parent.Right = newNode;
60.         newNode.orientation = true; //
61.     } else
62.     {
63.         parent.Left = newNode;
64.         newNode.orientation = false; //
65.     }
66.
67.     return newNode;
68. }
69.
70. boolean equal(double[] x1, double[] x2, int dim)
71. {
72.     for (int k = 0; k < dim; k++)
73.     {
74.         if (x1[k] != x2[k])
75.             return false;
76.     }
77.
78.     return true;
79. }
80.
81. double distance2(double[] x1, double[] x2, int dim)
82. {
83.     double S = 0;
84.     for (int k = 0; k < dim; k++)
85.         S += (x1[k] - x2[k]) * (x1[k] - x2[k]);
86.     return S;
87. }
88.
89.
90.class KD3DTree
91.
92.     KD3DNode Root;
93.
94.     int TimeStart, TimeFinish;
95.     int CounterFreq;
96.
97.     double d_min;
98.     KD3DNode nearest_neighbour;
99.
100.    int KD_id;
101.
102.    int nList;
103.
104.    KD3DNode CheckedNodes[];
105.    int checked_nodes;
106.    KD3DNode List[];
107.
108.    double x_min[], x_max[];
109.    boolean max_boundary[], min_boundary[];
110.    int n_boundary;
111.
112.    public KD3DTree(int i)
113.    {
114.        Root = null;

```

```

115.     KD_id = 1;
116.     nList = 0;
117.     List = new KD3DNode[i];
118.     CheckedNodes = new KD3DNode[i];
119.     max_boundary = new boolean[3];
120.     min_boundary = new boolean[3];
121.     x_min = new double[3];
122.     x_max = new double[3];
123. }
124.
125. public boolean add(double[] x)
126. {
127.     if (nList >= 2000000 - 1)
128.         return false; // can't add more points
129.
130.     if (Root == null)
131.     {
132.         Root = new KD3DNode(x, 0);
133.         Root.id = KD_id++;
134.         List[nList++] = Root;
135.     } else
136.     {
137.         KD3DNode pNode;
138.         if ((pNode = Root.Insert(x)) != null)
139.         {
140.             pNode.id = KD_id++;
141.             List[nList++] = pNode;
142.         }
143.     }
144.
145.     return true;
146. }
147.
148. public KD3DNode find_nearest(double[] x)
149. {
150.     if (Root == null)
151.         return null;
152.
153.     checked_nodes = 0;
154.     KD3DNode parent = Root.FindParent(x);
155.     nearest_neighbour = parent;
156.     d_min = Root.distance2(x, parent.x, 3);
157. ;
158.
159.     if (parent.equal(x, parent.x, 3) == true)
160.         return nearest_neighbour;
161.
162.     search_parent(parent, x);
163.     uncheck();
164.
165.     return nearest_neighbour;
166. }
167.
168. public void check_subtree(KD3DNode node, double[] x)
169. {
170.     if ((node == null) || node.checked)
171.         return;
172.
173.     CheckedNodes[checked_nodes++] = node;
174.     node.checked = true;
175.     set_bounding_cube(node, x);
176.
177.     int dim = node.axis;
178.     double d = node.x[dim] - x[dim];
179.
180.     if (d * d > d_min)

```

```

181.     {
182.         if (node.x[dim] > x[dim])
183.             check_subtree(node.Left, x);
184.         else
185.             check_subtree(node.Right, x);
186.     } else
187.     {
188.         check_subtree(node.Left, x);
189.         check_subtree(node.Right, x);
190.     }
191. }
192.
193. public void set_bounding_cube(KD3DNode node, double[] x)
194. {
195.     if (node == null)
196.         return;
197.     int d = 0;
198.     double dx;
199.     for (int k = 0; k < 3; k++)
200.     {
201.         dx = node.x[k] - x[k];
202.         if (dx > 0)
203.         {
204.             dx *= dx;
205.             if (!max_boundary[k])
206.             {
207.                 if (dx > x_max[k])
208.                     x_max[k] = dx;
209.                 if (x_max[k] > d_min)
210.                 {
211.                     max_boundary[k] = true;
212.                     n_boundary++;
213.                 }
214.             }
215.         } else
216.         {
217.             dx *= dx;
218.             if (!min_boundary[k])
219.             {
220.                 if (dx > x_min[k])
221.                     x_min[k] = dx;
222.                 if (x_min[k] > d_min)
223.                 {
224.                     min_boundary[k] = true;
225.                     n_boundary++;
226.                 }
227.             }
228.         }
229.         d += dx;
230.         if (d > d_min)
231.             return;
232.     }
233. }
234.
235. if (d < d_min)
236. {
237.     d_min = d;
238.     nearest_neighbour = node;
239. }
240. }
241.
242. public KD3DNode search_parent(KD3DNode parent, double[] x)
243. {
244.     for (int k = 0; k < 3; k++)
245.     {
246.         x_min[k] = x_max[k] = 0;

```

```

247.     max_boundary[k] = min_boundary[k] = false; //
248. }
249. n_boundary = 0;
250.
251. KD3DNode search_root = parent;
252. while (parent != null && (n_boundary != 3 * 3))
253. {
254.     check_subtree(parent, x);
255.     search_root = parent;
256.     parent = parent.Parent;
257. }
258.
259. return search_root;
260. }
261.
262. public void uncheck()
263. {
264.     for (int n = 0; n < checked_nodes; n++)
265.         CheckedNodes[n].checked = false;
266. }
267.
268. public void inorder()
269. {
270.     inorder(Root);
271. }
272.
273. private void inorder(KD3DNode root)
274. {
275.     if (root != null)
276.     {
277.         inorder(root.Left);
278.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ", "
279.                         + root.x[2] + ")");
280.         inorder(root.Right);
281.     }
282. }
283.
284. public void preorder()
285. {
286.     preorder(Root);
287. }
288.
289. private void preorder(KD3DNode root)
290. {
291.     if (root != null)
292.     {
293.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ", "
294.                         + root.x[2] + ")");
295.         inorder(root.Left);
296.         inorder(root.Right);
297.     }
298. }
299.
300. public void postorder()
301. {
302.     postorder(Root);
303. }
304.
305. private void postorder(KD3DNode root)
306. {
307.     if (root != null)
308.     {
309.         inorder(root.Left);
310.         inorder(root.Right);
311.         System.out.print("(" + root.x[0] + ", " + root.x[1] + ", "
312.                         + root.x[2] + ")");
}

```

```

313.     }
314. }
315.
316. public void search(double x, double y, double z)
317. {
318.     search(Root, x, y, z);
319. }
320.
321. private void search(KD3DNode root, double x, double y, double z)
322. {
323.     if (root != null)
324.     {
325.         search(root.Left, x, y, z);
326.         if (x == root.x[0] && y == root.x[1] && z == root.x[2])
327.             System.out.print("True (" + root.x[0] + ", " + root.x[1] + ", "
328.                             + root.x[2] + ")");
329.         search(root.Right, x, y, z);
330.     }
331. }
332. }
333.
334. public class KD3D_Search
335. {
336.     public static void main(String args[]) throws IOException
337.     {
338.         int numpoints = 5;
339.         Scanner sc = new Scanner(System.in);
340.         KD3DTTree kdt = new KD3DTTree(numpoints);
341.         double x[] = new double[3];
342.
343.         x[0] = 0.0;
344.         x[1] = 0.0;
345.         x[2] = 0.0;
346.         kdt.add(x);
347.
348.         x[0] = 3.3;
349.         x[1] = 1.5;
350.         x[2] = 4.0;
351.         kdt.add(x);
352.
353.         x[0] = 4.7;
354.         x[1] = 11.1;
355.         x[2] = 2.3;
356.         kdt.add(x);
357.
358.         x[0] = 5.0;
359.         x[1] = 12.3;
360.         x[2] = 5.7;
361.         kdt.add(x);
362.
363.         x[0] = 5.1;
364.         x[1] = 1.2;
365.         x[2] = 4.2;
366.         kdt.add(x);
367.
368.         System.out.println("Enter the co-ordinates of the point: <x> <y> <z>");
369.         double x1 = sc.nextDouble();
370.         double y1 = sc.nextDouble();
371.         double z1 = sc.nextDouble();
372.
373.         kdt.search(x1, y1, z1);
374.
375.         System.out.println("\nInorder of 2D Kd tree: ");
376.         kdt.inorder();
377.
378.         System.out.println("\nPreorder of 2D Kd tree: ");

```

```
379.     kdt.preorder();
380.
381.     System.out.println("\npostorder of 2D Kd tree: ");
382.     kdt.postorder();
383.     sc.close();
384.   }
385. }
```

Output:

advertisement

```
$ javac KD3D_Search.java
$ java KD3D_Search
```

```
Enter the co-ordinates of the point: <x> <y> <z>
5.1 1.2 4.2
True (5.1, 1.2, 4.2)
Inorder of 2D Kd tree:
(0.0, 0.0, 0.0) (5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7)
Preorder of 2D Kd tree:
(0.0, 0.0, 0.0) (5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7)
postorder of 2D Kd tree:
(5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7) (0.0, 0.0, 0.0)
```

```
Enter the co-ordinates of the point: <x> <y> <z>
5.1 5.2 5.3
False
Inorder of 2D Kd tree:
(0.0, 0.0, 0.0) (5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7)
Preorder of 2D Kd tree:
(0.0, 0.0, 0.0) (5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7)
postorder of 2D Kd tree:
(5.1, 1.2, 4.2) (3.3, 1.5, 4.0) (4.7, 11.1, 2.3) (5.0, 12.3, 5.7) (0.0, 0.0, 0.0)
```

280. Java Program to Implement First Fit Decreasing for 1-D Objects and M Bins

[« Prev](#)

[Next »](#)

This is a Java Program to implement First fit Decreasing Bin Packing algorithm. The First Fit Decreasing (FFD) strategy, operates by first sorting the items to be inserted in decreasing order by their sizes, and then inserting each item into the first bin in the list with sufficient remaining space.

Here is the source code of the Java Program to Implement First Fit Decreasing for 1-D Objects and M Bins. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement first fit decreasing for 1D objects using M bins
2. import java.util.Scanner;
3.
4. public class First_Fit_Decreasing_Bin_Packing
5. {
6.     public static void main(String args[])
7.     {
8.         System.out
9.             .println("BIN - PACKING Algorithm 1D Objects(First Fit Decreasing)");
10.            System.out.println("Enter the number of items in Set: ");
11.            Scanner sc = new Scanner(System.in);
12.            int n = sc.nextInt();
13.            System.out.println("Enter " + n + " items:");
14.            int[] a = new int[n];
15.            for (int i = 0; i < n; i++)
16.                a[i] = sc.nextInt();
17.            System.out.println("Enter the bin size: ");
18.            int size = sc.nextInt();
19.            int[] sequence = sort(a);
20.            binPacking(sequence, size, n);
21.            sc.close();
22.    }
23.
24.    public static void binPacking(int[] a, int size, int n)
25.    {
26.        int binCount = 0;
27.        int[] binValues = new int[n];
28.        for (int i = 0; i < binValues.length; i++)
29.            binValues[i] = size;
30.
31.        for (int i = 0; i < n; i++)
32.            for (int j = 0; j < binValues.length; j++)
33.            {
34.                if (binValues[j] - a[i] >= 0)
35.                {
36.                    binValues[j] -= a[i];
37.                    break;
38.                }
39.            }
40.
41.        for (int i = 0; i < binValues.length; i++)
42.            if (binValues[i] != size)
43.                binCount++;
44.
45.        System.out
46.            .println("Number of bins required using first fit decreasing algorithm is:"
47.                    + binCount);
48.    }
49.
50.    static int[] sort(int[] sequence)
```

```
51. {
52.     // Bubble Sort descending order
53.     for (int i = 0; i < sequence.length; i++)
54.         for (int j = 0; j < sequence.length - 1; j++)
55.             if (sequence[j] < sequence[j + 1])
56.             {
57.                 sequence[j] = sequence[j] + sequence[j + 1];
58.                 sequence[j + 1] = sequence[j] - sequence[j + 1];
59.                 sequence[j] = sequence[j] - sequence[j + 1];
60.             }
61.     return sequence;
62. }
63. }
```

Output:

advertisement

```
$ javac First_Fit_Decreasing_Bin_Packing.java
$ java First_Fit_Decreasing_Bin_Packing
```

BIN - PACKING Algorithm for 1D Objects(First Fit Decreasing)

Enter the number of items in Set:

9

Enter 9 items:

4

1

2

5

3

2

3

6

3

Enter the bin size:

6

Number of bins required using first fit decreasing algorithm is:5

281. Java Program to Implement Lloyd's Algorithm

[« Prev](#)

[Next »](#)

This is a Java Program to implement Lloyd's Algorithm. The LBG-algorithm or Lloyd's algorithm allows clustering of vectors of any dimension. This is helpful for example for image classification when using the SIFT or SURF algorithms. It might be also useful if you want to cluster a large amount of points on a map.

Here is the source code of the Java Program to Implement Lloyd's Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement Lloyd's Algorithm
2. import java.util.ArrayList;
3.
4. public class GenLloyd
5. {
6.     protected double[][] samplePoints;
7.     protected double[][] clusterPoints;
8.
9.     int[] pointApproxIndices;
10.    int pointDimension = 0;
11.    protected double epsilon = 0.0005;
12.    protected double avgDistortion = 0.0;
13.
14. /**
15. * Create Generalized Lloyd object with an array of sample points
16. */
17. public GenLloyd(double[][] samplePoints)
18. {
19.     this.setSamplePoints(samplePoints);
20. }
21.
22. /**
23. * Return epsilon parameter (accuracy)
24. */
25. public double getEpsilon()
26. {
27.     return epsilon;
28. }
29.
30. /**
31. * Set epsilon parameter (accuracy). Should be a small number 0.0 < epsilon
32. * < 0.1
33. */
34. public void setEpsilon(double epsilon)
35. {
36.     this.epsilon = epsilon;
37. }
38.
39. /**
40. * Set array of sample points
41. */
42. public void setSamplePoints(double[][] samplePoints)
43. {
44.     if (samplePoints.length > 0)
45.     {
46.         this.samplePoints = samplePoints;
47.         this.pointDimension = samplePoints[0].length;
48.     }
49. }
```

```

51. /**
52. * Get array of sample points
53. */
54. public double[][] getSamplePoints()
55. {
56.     return samplePoints;
57. }
58.
59. /**
60. * Get calculated cluster points. <numClusters> cluster points will be
61. * calculated and returned
62. */
63. public double[][] getClusterPoints(int numClusters)
64. {
65.     this.calcClusters(numClusters);
66.
67.     return clusterPoints;
68. }
69.
70. protected void calcClusters(int numClusters)
71. {
72.     // initialize with first cluster
73.     clusterPoints = new double[1][pointDimension];
74.
75.     double[] newClusterPoint = initializeClusterPoint(samplePoints);
76.     clusterPoints[0] = newClusterPoint;
77.
78.     if (numClusters > 1)
79.     {
80.         // calculate initial average distortion
81.         avgDistortion = 0.0;
82.         for (double[] samplePoint : samplePoints)
83.         {
84.             avgDistortion += calcDist(samplePoint, newClusterPoint);
85.         }
86.
87.         avgDistortion /= (double) (samplePoints.length * pointDimension);
88.
89.         // set up array of point approximation indices
90.         pointApproxIndices = new int[samplePoints.length];
91.
92.         // split the clusters
93.         int i = 1;
94.         do
95.         {
96.             i = splitClusters();
97.         } while (i < numClusters);
98.     }
99. }
100.
101. protected int splitClusters()
102. {
103.     int newClusterPointSize = 2;
104.     if (clusterPoints.length != 1)
105.     {
106.         newClusterPointSize = clusterPoints.length * 2;
107.     }
108.
109.     // split clusters
110.     double[][] newClusterPoints = new double[newClusterPointSize][pointDimension];
111.     int newClusterPointIdx = 0;
112.     for (double[] clusterPoint : clusterPoints)
113.     {
114.         newClusterPoints[newClusterPointIdx] = createNewClusterPoint(
115.             clusterPoint, -1);
116.         newClusterPoints[newClusterPointIdx + 1] = createNewClusterPoint(

```

```

117.         clusterPoint, +1);
118.
119.     newClusterPointIdx += 2;
120. }
121.
122. clusterPoints = newClusterPoints;
123.
124. // iterate to approximate cluster points
125. // int iteration = 0;
126. double curAvgDistortion = 0.0;
127. do
128. {
129.     curAvgDistortion = avgDistortion;
130.
131.     //find the min values
132.     for (int pointIdx = 0; pointIdx < samplePoints.length; pointIdx++)
133.     {
134.         double minDist = Double.MAX_VALUE;
135.         for (int clusterPointIdx = 0; clusterPointIdx < clusterPoints.length; clusterPointIdx++)
136.         {
137.             double newMinDist = calcDist(samplePoints[pointIdx],
138.                 clusterPoints[clusterPointIdx]);
139.             if (newMinDist < minDist)
140.             {
141.                 minDist = newMinDist;
142.                 pointApproxIndices[pointIdx] = clusterPointIdx;
143.             }
144.         }
145.     }
146.
147.     // update codebook
148.     for (int clusterPointIdx = 0; clusterPointIdx < clusterPoints.length; clusterPointIdx++)
149.     {
150.         double[] newClusterPoint = new double[pointDimension];
151.         int num = 0;
152.         for (int pointIdx = 0; pointIdx < samplePoints.length; pointIdx++)
153.         {
154.             if (pointApproxIndices[pointIdx] == clusterPointIdx)
155.             {
156.                 addPointValues(newClusterPoint, samplePoints[pointIdx]);
157.                 num++;
158.             }
159.         }
160.
161.         if (num > 0)
162.         {
163.             multiplyPointValues(newClusterPoint, 1.0 / (double) num);
164.             clusterPoints[clusterPointIdx] = newClusterPoint;
165.         }
166.     }
167.
168.     // update average distortion
169.     avgDistortion = 0.0;
170.     for (int pointIdx = 0; pointIdx < samplePoints.length; pointIdx++)
171.     {
172.         avgDistortion += calcDist(samplePoints[pointIdx],
173.             clusterPoints[pointApproxIndices[pointIdx]]);
174.     }
175.
176.     avgDistortion /= (double) (samplePoints.length * pointDimension);
177.
178. } while (((curAvgDistortion - avgDistortion) / curAvgDistortion) > epsilon);
179.
180. return clusterPoints.length;
181. }
182.

```

```

183. protected double[] initializeClusterPoint(double[][] pointsInCluster)
184. {
185.     // calculate point sum
186.     double[] clusterPoint = new double[pointDimension];
187.     for (int numPoint = 0; numPoint < pointsInCluster.length; numPoint++)
188.     {
189.         addPointValues(clusterPoint, pointsInCluster[numPoint]);
190.     }
191.
192.     // calculate average
193.     multiplyPointValues(clusterPoint, 1.0 / (double) pointsInCluster.length);
194.
195.     return clusterPoint;
196. }
197.
198. protected double[] createNewClusterPoint(double[] clusterPoint,
199.                                         int epsilonFactor)
200. {
201.     double[] newClusterPoint = new double[pointDimension];
202.     addPointValues(newClusterPoint, clusterPoint);
203.     multiplyPointValues(newClusterPoint, 1.0 + (double) epsilonFactor
204.                           * epsilon);
205.
206.     return newClusterPoint;
207. }
208.
209. protected double calcDist(double[] v1, double[] v2)
210. {
211.     double distSum = 0.0;
212.     for (int pointIdx = 0; pointIdx < v1.length; pointIdx++)
213.     {
214.         double absDist = Math.abs(v1[pointIdx] - v2[pointIdx]);
215.         distSum += absDist * absDist;
216.     }
217.
218.     return distSum;
219. }
220.
221. protected void addPointValues(double[] v1, double[] v2)
222. {
223.     for (int pointIdx = 0; pointIdx < v1.length; pointIdx++)
224.     {
225.         v1[pointIdx] += v2[pointIdx];
226.     }
227. }
228.
229. protected void multiplyPointValues(double[] v1, double f)
230. {
231.     for (int pointIdx = 0; pointIdx < v1.length; pointIdx++)
232.     {
233.         v1[pointIdx] *= f;
234.     }
235. }
236.
237. public static void main(String[] args)
238. {
239.     ArrayList<double[]> points = new ArrayList<double[]>();
240.
241.     // points.add(arrayOf(-1.5, -1.5));
242.     points.add(arrayOf(-1.5, 2.0, 5.0));
243.     points.add(arrayOf(-2.0, -2.0, 0.0));
244.     points.add(arrayOf(1.0, 1.0, 2.0));
245.     points.add(arrayOf(1.5, 1.5, 1.2));
246.     points.add(arrayOf(1.0, 2.0, 5.6));
247.     points.add(arrayOf(1.0, -2.0, -2.0));
248.     points.add(arrayOf(1.0, -3.0, -2.0));

```

```
249.     points.add(arrayOf(1.0, -2.5, -4.5));
250.
251.     GenLloyd gl = new GenLloyd(points.toArray(new double[points.size()][3]));
252.
253.     double[][] results = gl.getClusterPoints(4);
254.     for (double[] point : results)
255.     {
256.         System.out.println("Cluster " + point[0] + ", " + point[1] + ", "
257.                             + point[2]);
258.     }
259. }
260.
261. private static double[] arrayOf(double x, double y, double z)
262. {
263.     double[] a = new double[3];
264.     a[0] = x;
265.     a[1] = y;
266.     a[2] = z;
267.
268.     return a;
269. }
270. }
```

Output:

advertisement

```
$ javac GenLloyd.java
$ java GenLloyd
```

```
Cluster -2.0, -2.0, 0.0
Cluster 1.0, -2.5, -2.833333333333333
Cluster 1.25, 1.25, 1.6
Cluster -0.25, 2.0, 5.3
```

282. Java Program to Implement Gift Wrapping Algorithm in Two Dimensions

[« Prev](#)

[Next »](#)

This is a Java Program to implement Gift Wrapping Algorithm. For the sake of simplicity, the description below assumes that the points are in general position, i.e., no three points are collinear. The algorithm may be easily modified to deal with collinearity, including the choice whether it should report only extreme points (vertices of the convex hull) or all points that lie on the convex hull.

Here is the source code of the Java Program to Implement Gift Wrapping Algorithm in Two Dimensions. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement gift warpping algorithm in 2 dimension
2. import java.util.Arrays;
3. import java.util.Comparator;
4. import java.util.Scanner;
5. import java.util.Stack;
6.
7. class Point2D implements Comparable<Point2D>
8. {
9.     public static final Comparator<Point2D> X_ORDER = new XOrder();
10.    public static final Comparator<Point2D> Y_ORDER = new YOrder();
11.    public static final Comparator<Point2D> R_ORDER = new ROrder();
12.    public final Comparator<Point2D> POLAR_ORDER = new PolarOrder();
13.    public final Comparator<Point2D> ATAN2_ORDER = new Atan2Order();
14.    public final Comparator<Point2D> DISTANCE_TO_ORDER = new DistanceToOrder();
15.
16.    private final double x; // x coordinate
17.    private final double y; // y coordinate
18.
19.    public Point2D(double x, double y)
20.    {
21.        if (Double.isInfinite(x) || Double.isInfinite(y))
22.            throw new IllegalArgumentException("Coordinates must be finite");
23.        if (Double.isNaN(x) || Double.isNaN(y))
24.            throw new IllegalArgumentException("Coordinates cannot be NaN");
25.        if (x == 0.0)
26.            x = 0.0; // convert -0.0 to +0.0
27.        if (y == 0.0)
28.            y = 0.0; // convert -0.0 to +0.0
29.        this.x = x;
30.        this.y = y;
31.    }
32.
33.    public double x()
34.    {
35.        return x;
36.    }
37.
38.    public double y()
39.    {
40.        return y;
41.    }
42.
43.    public double r()
44.    {
45.        return Math.sqrt(x * x + y * y);
46.    }
47.
48.    public double theta()
49.    {
```

```
50.     return Math.atan2(y, x);
51. }
52.
53. private double angleTo(Point2D that)
54. {
55.     double dx = that.x - this.x;
56.     double dy = that.y - this.y;
57.     return Math.atan2(dy, dx);
58. }
59.
60. public static int ccw(Point2D a, Point2D b, Point2D c)
61. {
62.     double area2 = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
63.     if (area2 < 0)
64.         return -1;
65.     else if (area2 > 0)
66.         return +1;
67.     else
68.         return 0;
69. }
70.
71. public static double area2(Point2D a, Point2D b, Point2D c)
72. {
73.     return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
74. }
75.
76. public double distanceTo(Point2D that)
77. {
78.     double dx = this.x - that.x;
79.     double dy = this.y - that.y;
80.     return Math.sqrt(dx * dx + dy * dy);
81. }
82.
83. public double distanceSquaredTo(Point2D that)
84. {
85.     double dx = this.x - that.x;
86.     double dy = this.y - that.y;
87.     return dx * dx + dy * dy;
88. }
89.
90. public int compareTo(Point2D that)
91. {
92.     if (this.y < that.y)
93.         return -1;
94.     if (this.y > that.y)
95.         return +1;
96.     if (this.x < that.x)
97.         return -1;
98.     if (this.x > that.x)
99.         return +1;
100.    return 0;
101. }
102.
103. private static class XOrder implements Comparator<Point2D>
104. {
105.     public int compare(Point2D p, Point2D q)
106.     {
107.         if (p.x < q.x)
108.             return -1;
109.         if (p.x > q.x)
110.             return +1;
111.         return 0;
112.     }
113. }
114.
115. private static class YOrder implements Comparator<Point2D>
```

```

116. {
117.     public int compare(Point2D p, Point2D q)
118.     {
119.         if (p.y < q.y)
120.             return -1;
121.         if (p.y > q.y)
122.             return +1;
123.         return 0;
124.     }
125. }
126.
127. private static class ROrder implements Comparator<Point2D>
128. {
129.     public int compare(Point2D p, Point2D q)
130.     {
131.         double delta = (p.x * p.x + p.y * p.y) - (q.x * q.x + q.y * q.y);
132.         if (delta < 0)
133.             return -1;
134.         if (delta > 0)
135.             return +1;
136.         return 0;
137.     }
138. }
139.
140. private class Atan2Order implements Comparator<Point2D>
141. {
142.     public int compare(Point2D q1, Point2D q2)
143.     {
144.         double angle1 = angleTo(q1);
145.         double angle2 = angleTo(q2);
146.         if (angle1 < angle2)
147.             return -1;
148.         else if (angle1 > angle2)
149.             return +1;
150.         else
151.             return 0;
152.     }
153. }
154.
155. private class PolarOrder implements Comparator<Point2D>
156. {
157.     public int compare(Point2D q1, Point2D q2)
158.     {
159.         double dx1 = q1.x - x;
160.         double dy1 = q1.y - y;
161.         double dx2 = q2.x - x;
162.         double dy2 = q2.y - y;
163.
164.         if (dy1 >= 0 && dy2 < 0)
165.             return -1; // q1 above; q2 below
166.         else if (dy2 >= 0 && dy1 < 0)
167.             return +1; // q1 below; q2 above
168.         else if (dy1 == 0 && dy2 == 0)
169.             { // 3-collinear and horizontal
170.                 if (dx1 >= 0 && dx2 < 0)
171.                     return -1;
172.                 else if (dx2 >= 0 && dx1 < 0)
173.                     return +1;
174.                 else
175.                     return 0;
176.             } else
177.                 return -ccw(Point2D.this, q1, q2); // both above or below
178.     }
179. }
180.
181. private class DistanceToOrder implements Comparator<Point2D>

```

```

182. {
183.     public int compare(Point2D p, Point2D q)
184.     {
185.         double dist1 = distanceSquaredTo(p);
186.         double dist2 = distanceSquaredTo(q);
187.         if (dist1 < dist2)
188.             return -1;
189.         else if (dist1 > dist2)
190.             return +1;
191.         else
192.             return 0;
193.     }
194. }
195.
196. public boolean equals(Object other)
197. {
198.     if (other == this)
199.         return true;
200.     if (other == null)
201.         return false;
202.     if (other.getClass() != this.getClass())
203.         return false;
204.     Point2D that = (Point2D) other;
205.     return this.x == that.x && this.y == that.y;
206. }
207.
208. public String toString()
209. {
210.     return "(" + x + ", " + y + ")";
211. }
212.
213. public int hashCode()
214. {
215.     int hashX = ((Double) x).hashCode();
216.     int hashY = ((Double) y).hashCode();
217.     return 31 * hashX + hashY;
218. }
219.
220. }
221.
222. public class Gift_Wrapping_Algorithm
223. {
224.     private Stack<Point2D> hull = new Stack<Point2D>();
225.
226.     public Gift_Wrapping_Algorithm(Point2D[] pts)
227.     {
228.
229.         // defensive copy
230.         int N = pts.length;
231.         Point2D[] points = new Point2D[N];
232.         for (int i = 0; i < N; i++)
233.             points[i] = pts[i];
234.         Arrays.sort(points);
235.
236.         Arrays.sort(points, 1, N, points[0].POLAR_ORDER);
237.
238.         hull.push(points[0]); // p[0] is first extreme point
239.         int k1;
240.         for (k1 = 1; k1 < N; k1++)
241.             if (!points[0].equals(points[k1]))
242.                 break;
243.         if (k1 == N)
244.             return; // all points equal
245.
246.         int k2;
247.         for (k2 = k1 + 1; k2 < N; k2++)

```

```

248.     if (Point2D.ccw(points[0], points[k1], points[k2]) != 0)
249.         break;
250.     hull.push(points[k2 - 1]); // points[k2-1] is second extreme point
251.
252.     for (int i = k2; i < N; i++)
253.     {
254.         Point2D top = hull.pop();
255.         while (Point2D.ccw(hull.peek(), top, points[i]) <= 0)
256.         {
257.             top = hull.pop();
258.         }
259.         hull.push(top);
260.         hull.push(points[i]);
261.     }
262.
263.     assert isConvex();
264. }
265.
266. public Iterable<Point2D> hull()
267. {
268.     Stack<Point2D> s = new Stack<Point2D>();
269.     for (Point2D p : hull)
270.         s.push(p);
271.     return s;
272. }
273.
274. private boolean isConvex()
275. {
276.     int N = hull.size();
277.     if (N <= 2)
278.         return true;
279.
280.     Point2D[] points = new Point2D[N];
281.     int n = 0;
282.     for (Point2D p : hull())
283.     {
284.         points[n++] = p;
285.     }
286.
287.     for (int i = 0; i < N; i++)
288.     {
289.         if (Point2D
290.             .ccw(points[i], points[(i + 1) % N], points[(i + 2) % N]) <= 0)
291.         {
292.             return false;
293.         }
294.     }
295.     return true;
296. }
297.
298. // test client
299. public static void main(String[] args)
300. {
301.     System.out.println("Gift Wrapping Algorithm");
302.     Scanner sc = new Scanner(System.in);
303.     System.out.println("Enter the number of points");
304.     int N = sc.nextInt();
305.
306.     Point2D[] points = new Point2D[N];
307.     System.out.println("Enter the coordinates of each points: <x> <y>");
308.     for (int i = 0; i < N; i++)
309.     {
310.         int x = sc.nextInt();
311.         int y = sc.nextInt();
312.         points[i] = new Point2D(x, y);
313.     }

```

```
314.     Gift_Wrapping_Algorithm graham = new Gift_Wrapping_Algorithm(points);
315.     System.out.println("The Wrapper covers following points: ");
316.     for (Point2D p : graham.hull())
317.         System.out.println(p);
318.
319.     sc.close();
320. }
321.
322.}
```

Output:

advertisement

```
$ javac Gift_Wrapping_Algorithm.java
$ java Gift_Wrapping_Algorithm
```

```
Gift Wrapping Algorithm
Enter the number of points
5
Enter the coordinates of each points: <x> <y>
12 10
23 24
10 6
5 3
3 1
The Wrapper covers following points:
(3.0, 1.0)
(10.0, 6.0)
(23.0, 24.0)
```

283. Java Program to Implement Graham Scan Algorithm to Find the Convex Hull

[« Prev](#)

[Next »](#)

This is a Java Program to implement Graham Scan Algorithm. Graham's scan is a method of computing the convex hull of a finite set of points in the plane with time complexity O(n log n).

Here is the source code of the Java Program to Implement Graham Scan Algorithm to Find the Convex Hull. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to implement Graham Scan Algorithm
2. import java.util.Arrays;
3. import java.util.Comparator;
4. import java.util.Scanner;
5. import java.util.Stack;
6.
7. class Point2D implements Comparable<Point2D>
8. {
9.     public static final Comparator<Point2D> X_ORDER = new XOrder();
10.    public static final Comparator<Point2D> Y_ORDER = new YOrder();
11.    public static final Comparator<Point2D> R_ORDER = new ROrder();
12.    public final Comparator<Point2D> POLAR_ORDER = new PolarOrder();
13.    public final Comparator<Point2D> ATAN2_ORDER = new Atan2Order();
14.    public final Comparator<Point2D> DISTANCE_TO_ORDER = new DistanceToOrder();
15.
16.    private final double x; // x coordinate
17.    private final double y; // y coordinate
18.
19.    public Point2D(double x, double y)
20.    {
21.        if (Double.isInfinite(x) || Double.isInfinite(y))
22.            throw new IllegalArgumentException("Coordinates must be finite");
23.        if (Double.isNaN(x) || Double.isNaN(y))
24.            throw new IllegalArgumentException("Coordinates cannot be NaN");
25.        if (x == 0.0)
26.            x = 0.0; // convert -0.0 to +0.0
27.        if (y == 0.0)
28.            y = 0.0; // convert -0.0 to +0.0
29.        this.x = x;
30.        this.y = y;
31.    }
32.
33.    public double x()
34.    {
35.        return x;
36.    }
37.
38.    public double y()
39.    {
40.        return y;
41.    }
42.
43.    public double r()
44.    {
45.        return Math.sqrt(x * x + y * y);
46.    }
47.
48.    public double theta()
49.    {
50.        return Math.atan2(y, x);
51.    }
```

```

52.
53.     private double angleTo(Point2D that)
54.     {
55.         double dx = that.x - this.x;
56.         double dy = that.y - this.y;
57.         return Math.atan2(dy, dx);
58.     }
59.
60.     public static int ccw(Point2D a, Point2D b, Point2D c)
61.     {
62.         double area2 = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
63.         if (area2 < 0)
64.             return -1;
65.         else if (area2 > 0)
66.             return +1;
67.         else
68.             return 0;
69.     }
70.
71.     public static double area2(Point2D a, Point2D b, Point2D c)
72.     {
73.         return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
74.     }
75.
76.     public double distanceTo(Point2D that)
77.     {
78.         double dx = this.x - that.x;
79.         double dy = this.y - that.y;
80.         return Math.sqrt(dx * dx + dy * dy);
81.     }
82.
83.     public double distanceSquaredTo(Point2D that)
84.     {
85.         double dx = this.x - that.x;
86.         double dy = this.y - that.y;
87.         return dx * dx + dy * dy;
88.     }
89.
90.     public int compareTo(Point2D that)
91.     {
92.         if (this.y < that.y)
93.             return -1;
94.         if (this.y > that.y)
95.             return +1;
96.         if (this.x < that.x)
97.             return -1;
98.         if (this.x > that.x)
99.             return +1;
100.        return 0;
101.    }
102.
103.    private static class XOrder implements Comparator<Point2D>
104.    {
105.        public int compare(Point2D p, Point2D q)
106.        {
107.            if (p.x < q.x)
108.                return -1;
109.            if (p.x > q.x)
110.                return +1;
111.            return 0;
112.        }
113.    }
114.
115.    private static class YOrder implements Comparator<Point2D>
116.    {
117.        public int compare(Point2D p, Point2D q)

```

```

118.     {
119.         if (p.y < q.y)
120.             return -1;
121.         if (p.y > q.y)
122.             return +1;
123.         return 0;
124.     }
125. }
126.
127. private static class ROrder implements Comparator<Point2D>
128. {
129.     public int compare(Point2D p, Point2D q)
130.     {
131.         double delta = (p.x * p.x + p.y * p.y) - (q.x * q.x + q.y * q.y);
132.         if (delta < 0)
133.             return -1;
134.         if (delta > 0)
135.             return +1;
136.         return 0;
137.     }
138. }
139.
140. private class Atan2Order implements Comparator<Point2D>
141. {
142.     public int compare(Point2D q1, Point2D q2)
143.     {
144.         double angle1 = angleTo(q1);
145.         double angle2 = angleTo(q2);
146.         if (angle1 < angle2)
147.             return -1;
148.         else if (angle1 > angle2)
149.             return +1;
150.         else
151.             return 0;
152.     }
153. }
154.
155. private class PolarOrder implements Comparator<Point2D>
156. {
157.     public int compare(Point2D q1, Point2D q2)
158.     {
159.         double dx1 = q1.x - x;
160.         double dy1 = q1.y - y;
161.         double dx2 = q2.x - x;
162.         double dy2 = q2.y - y;
163.
164.         if (dy1 >= 0 && dy2 < 0)
165.             return -1; // q1 above; q2 below
166.         else if (dy2 >= 0 && dy1 < 0)
167.             return +1; // q1 below; q2 above
168.         else if (dy1 == 0 && dy2 == 0)
169.             { // 3-collinear and horizontal
170.                 if (dx1 >= 0 && dx2 < 0)
171.                     return -1;
172.                 else if (dx2 >= 0 && dx1 < 0)
173.                     return +1;
174.                 else
175.                     return 0;
176.             } else
177.                 return -ccw(Point2D.this, q1, q2); // both above or below
178.     }
179. }
180.
181. private class DistanceToOrder implements Comparator<Point2D>
182. {
183.     public int compare(Point2D p, Point2D q)

```

```

184.     {
185.         double dist1 = distanceSquaredTo(p);
186.         double dist2 = distanceSquaredTo(q);
187.         if (dist1 < dist2)
188.             return -1;
189.         else if (dist1 > dist2)
190.             return +1;
191.         else
192.             return 0;
193.     }
194. }
195.
196. public boolean equals(Object other)
197. {
198.     if (other == this)
199.         return true;
200.     if (other == null)
201.         return false;
202.     if (other.getClass() != this.getClass())
203.         return false;
204.     Point2D that = (Point2D) other;
205.     return this.x == that.x && this.y == that.y;
206. }
207.
208. public String toString()
209. {
210.     return "(" + x + ", " + y + ")";
211. }
212.
213. public int hashCode()
214. {
215.     int hashX = ((Double) x).hashCode();
216.     int hashY = ((Double) y).hashCode();
217.     return 31 * hashX + hashY;
218. }
219.
220. }
221.
222. public class GrahamScan
223. {
224.     private Stack<Point2D> hull = new Stack<Point2D>();
225.
226.     public GrahamScan(Point2D[] pts)
227.     {
228.
229.         // defensive copy
230.         int N = pts.length;
231.         Point2D[] points = new Point2D[N];
232.         for (int i = 0; i < N; i++)
233.             points[i] = pts[i];
234.         Arrays.sort(points);
235.
236.         Arrays.sort(points, 1, N, points[0].POLAR_ORDER);
237.
238.         hull.push(points[0]); // p[0] is first extreme point
239.         int k1;
240.         for (k1 = 1; k1 < N; k1++)
241.             if (!points[0].equals(points[k1]))
242.                 break;
243.         if (k1 == N)
244.             return; // all points equal
245.
246.         int k2;
247.         for (k2 = k1 + 1; k2 < N; k2++)
248.             if (Point2D.ccw(points[0], points[k1], points[k2]) != 0)
249.                 break;

```

```

250.     hull.push(points[k2 - 1]); //points[k2-1] is second extreme point
251.
252.    for (int i = k2; i < N; i++)
253.    {
254.        Point2D top = hull.pop();
255.        while (Point2D.ccw(hull.peek(), top, points[i]) <= 0)
256.        {
257.            top = hull.pop();
258.        }
259.        hull.push(top);
260.        hull.push(points[i]);
261.    }
262.
263.    assert isConvex();
264. }
265.
266. public Iterable<Point2D> hull()
267. {
268.     Stack<Point2D> s = new Stack<Point2D>();
269.     for (Point2D p : hull)
270.         s.push(p);
271.     return s;
272. }
273.
274. private boolean isConvex()
275. {
276.     int N = hull.size();
277.     if (N <= 2)
278.         return true;
279.
280.     Point2D[] points = new Point2D[N];
281.     int n = 0;
282.     for (Point2D p : hull())
283.     {
284.         points[n++] = p;
285.     }
286.
287.     for (int i = 0; i < N; i++)
288.     {
289.         if (Point2D
290.             .ccw(points[i], points[(i + 1) % N], points[(i + 2) % N]) <= 0)
291.         {
292.             return false;
293.         }
294.     }
295.     return true;
296. }
297.
298. // test client
299. public static void main(String[] args)
300. {
301.     System.out.println("Graham Scan Test");
302.     Scanner sc = new Scanner(System.in);
303.     System.out.println("Enter the number of points");
304.     int N = sc.nextInt();
305.
306.     Point2D[] points = new Point2D[N];
307.     System.out.println("Enter the coordinates of each points: <x> <y>");
308.     for (int i = 0; i < N; i++)
309.     {
310.         int x = sc.nextInt();
311.         int y = sc.nextInt();
312.         points[i] = new Point2D(x, y);
313.     }
314.     GrahamScan graham = new GrahamScan(points);
315.     System.out.println("The convex hull consists of following points: ");

```

```
316.     for (Point2D p : graham.hull())
317.         System.out.println(p);
318.
319.     sc.close();
320. }
321.
322.}
```

Output:

advertisement

```
$ javac GrahamScan.java
$ java GrahamScan

Graham Scan Test
Enter the number of points
5
Enter the coordinates of each points: <x> <y>
1 2
2 3
4 5
20 10
6 4
The convex hull consists of following points:
(1.0, 2.0)
(6.0, 4.0)
(20.0, 10.0)
(4.0, 5.0)
```

```
Graham Scan Test
Enter the number of points
5
Enter the coordinates of each points: <x> <y>
1 2
2 3
3 4
4 5
5 6
The convex hull consists of following points:
(1.0, 2.0)
(5.0, 6.0)
```

284. Java Program to Use Above Below Primitive to Test Whether Two Lines Intersect

[« Prev](#)

[Next »](#)

This is a Java Program to whether two lines intersect or not. The above-below primitive can be used to test whether a line intersects a line segment. It does iff one endpoint of the segment is to the left of the line and the other is to the right. Segment intersection is similar but more complicated, and we refer you to implementations described below. The decision whether two segments intersect if they share an endpoint depends upon your application and is representative of the problems of degeneracy.

Here is the source code of the Java Program to Use Above Below Primitive to Test Whether Two Lines Intersect. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find whether two lines intersect or not using above and below primitive
2. import java.util.Random;
3.
4. public class Line_Intersection
5. {
6.     public static void main(String args[])
7.     {
8.         Random random = new Random();
9.
10.        int x1, x2, y1, y2;
11.        x1 = random.nextInt(10);
12.        x2 = random.nextInt(10);
13.        y1 = random.nextInt(10);
14.        y2 = random.nextInt(10);
15.
16.        System.out.println("The Equation of the 1st line is : (" + (y2 - y1)
17.                           + "x+" + (x1 - x2) + "y+" + (x2 * y1 - x1 * y2) + ")= 0");
18.
19.        int p1, p2, q1, q2;
20.        p1 = random.nextInt(10);
21.        p2 = random.nextInt(10);
22.        q1 = random.nextInt(10);
23.        q2 = random.nextInt(10);
24.
25.        System.out.println("The Equation of the 2nd line is : (" + (q2 - q1)
26.                           + "x+" + (p1 - p2) + "y+" + (p2 * q1 - p1 * q2) + ")= 0");
27.
28.        int s1 = (y2 - y1) * p1 + (x1 - x2) * q1 + (x2 * y1 - x1 * y2);
29.        if (s1 < 0)
30.        {
31.            int s2 = (y2 - y1) * p2 + (x1 - x2) * q2 + (x2 * y1 - x1 * y2);
32.            if (s2 >= 0)
33.                System.out.println("The lines intersect");
34.            else if (s2 < 0)
35.                System.out.println("The lines doesn't intersect");
36.
37.        }
38.        else if (s1 > 0)
39.        {
40.            int s2 = (y2 - y1) * p2 + (x1 - x2) * q2 + (x2 * y1 - x1 * y2);
41.            if (s2 <= 0)
42.                System.out.println("The lines intersect");
43.            else if (s2 > 0)
44.                System.out.println("The lines doesn't intersect");
45.        }
46.        else
47.            System.out.println("The point lies on the line");
```

```
48. }  
49. }
```

Output:

advertisement

```
$ javac Line_Intersection.java  
$ java Line_Intersection
```

The Equation of the line is : $(2)x + (9)y + (-63) = 0$
The Equation of the line is : $(-4)x + (2)y + (-4) = 0$
The lines doesn't intersect

The Equation of the line is : $(-4)x + (-3)y + (43) = 0$
The Equation of the line is : $(-1)x + (-8)y + (73) = 0$
The lines intersect

285. Java Program to Find Nearest Neighbor Using Linear Search

[« Prev](#)

[Next »](#)

This is a Java Program to find nearest neighbor using linear search. The simplest solution to the NNS problem is to compute the distance from the query point to every other point in the database, keeping track of the “best so far”. This algorithm, sometimes referred to as the naive approach, has a running time of $O(Nd)$ where N is the cardinality of S and d is the dimensionality of M . There are no search data structures to maintain, so linear search has no space complexity beyond the storage of the database. Naive search can, on average, outperform space partitioning approaches on higher dimensional spaces.

Here is the source code of the Java Program to Find Nearest Neighbor Using Linear Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find nearest neighbor using a simple linear search
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Linear_Search_Nearest
6. {
7.     public static void main(String args[])
8.     {
9.         Random random = new Random();
10.        Scanner sc = new Scanner(System.in);
11.        int datasetSize = 10;
12.        double[][] data = new double[10][2];
13.
14.        for (int i = 0; i < datasetSize; i++)
15.            for (int j = 0; j < 2; j++)
16.                data[i][j] = random.nextDouble() * 10;
17.
18.        System.out.println("Randomly generated Data set: ");
19.        for (int i = 0; i < datasetSize; i++)
20.            for (int j = 0; j < 2; j++)
21.                System.out.println(data[i][j] + " , " + data[i][j]);
22.
23.        System.out.println();
24.        System.out.println("Enter the co-ordinates of the point: <x><y>");
25.        double x = sc.nextDouble();
26.        double y = sc.nextDouble();
27.
28.        double xmin = data[0][0], ymin = data[0][1], xclose = 0, yclose = 0;
29.        for (int i = 0; i < datasetSize; i++)
30.        {
31.            if (Math.abs(data[i][0] - x) < xmin)
32.            {
33.                xmin = data[i][0] - x;
34.                xclose = data[i][0];
35.            }
36.        }
37.
38.        for (int i = 0; i < datasetSize; i++)
39.        {
40.            if (Math.abs(data[i][1] - y) < ymin)
41.            {
42.                ymin = data[i][1] - x;
43.                yclose = data[i][1];
44.            }
45.        }
46.
47.        System.out.println("The nearest neighbor is : (" + xclose + ", "
```

```
48.           + yclose + ")");
49.
50.       sc.close();
51.   }
52.}
```

Output:

advertisement

```
$ javac Linear_Search_Nearest.java
$ java Linear_Search_Nearest
```

Randomly generated Data set:

```
3.171455377670047 ,1.052119263026371
3.949033565232699 ,8.565344250655025
0.0208421026579253 ,5.963319480178625
5.9198056196163495 ,4.424992495072658
6.083654323496389 ,2.592835352360611
5.996752857974297 ,2.1046723166354777
3.165362843381636 ,5.1640243122381415
4.175425572150399 ,2.965443123350698
8.734700795907905 ,3.3650152184786064
5.5317982332184235 ,1.5076066489140683
```

Enter the co-ordinates of the point: <x> <y>

1 2

The nearest neighbor is : (0.0208421026579253, 1.052119263026371)

286. Java Program to Perform Polygon Containment Test

[« Prev](#)

[Next »](#)

This is a Java Program to check whether a polygon contains another polygon or not. The class returns true if one polygon contains another, false otherwise.

Here is the source code of the Java Program to Perform Polygon Containment Test. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to test whether a polygon is inside another polygon
2. class Point
3. {
4.     int x, y;
5.
6.     Point()
7.     {
8.     }
9.
10.    Point(int p, int q)
11.    {
12.        x = p;
13.        y = q;
14.    }
15.}
16.
17. public class Position_Point_WRT_Polygon
18. {
19.
20.     public static boolean onSegment(Point p, Point q, Point r)
21.     {
22.         if (q.x <= Math.max(p.x, r.x) && q.x >= Math.min(p.x, r.x)
23.             && q.y <= Math.max(p.y, r.y) && q.y >= Math.min(p.y, r.y))
24.             return true;
25.         return false;
26.     }
27.
28.     public static int orientation(Point p, Point q, Point r)
29.     {
30.         int val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) * (r.y - q.y);
31.
32.         if (val == 0)
33.             return 0;
34.         return (val > 0) ? 1 : 2;
35.     }
36.
37.     public static boolean doIntersect(Point p1, Point q1, Point p2, Point q2)
38.     {
39.
40.         int o1 = orientation(p1, q1, p2);
41.         int o2 = orientation(p1, q1, q2);
42.         int o3 = orientation(p2, q2, p1);
43.         int o4 = orientation(p2, q2, q1);
44.
45.         if (o1 != o2 && o3 != o4)
46.             return true;
47.
48.         if (o1 == 0 && onSegment(p1, p2, q1))
49.             return true;
50.
51.         if (o2 == 0 && onSegment(p1, q2, q1))
```

```

52.     return true;
53.
54.     if (o3 == 0 && onSegment(p2, p1, q2))
55.         return true;
56.
57.     if (o4 == 0 && onSegment(p2, q1, q2))
58.         return true;
59.
60.     return false;
61. }
62.
63. public static boolean isInside(Point polygon[], int n, Point p)
64. {
65.     int INF = 10000;
66.     if (n < 3)
67.         return false;
68.
69.     Point extreme = new Point(INF, p.y);
70.
71.     int count = 0, i = 0;
72.     do
73.     {
74.         int next = (i + 1) % n;
75.         if (doIntersect(polygon[i], polygon[next], p, extreme))
76.         {
77.             if (orientation(polygon[i], p, polygon[next]) == 0)
78.                 return onSegment(polygon[i], p, polygon[next]);
79.
80.             count++;
81.         }
82.         i = next;
83.     } while (i != 0);
84.
85.     return (count & 1) == 1 ? true : false;
86. }
87.
88. public static Boolean DoesPolygonContainPolygon(Point[] p1, Point[] p2)
89. {
90.     Point p;
91.     for (int i = 0; i < p2.length; i++)
92.     {
93.         p = new Point(p2[i].x, p2[i].y);
94.         if (!isInside(p1, p1.length, p))
95.             return false;
96.     }
97.     return true;
98. }
99.
100. public static void main(String args[])
101. {
102.     Point polygon1[] = { new Point(0, 0), new Point(10, 0),
103.                          new Point(10, 10), new Point(0, 10) };
104.
105.     Point polygon2[] = { new Point(0, 0), new Point(15, 5), new Point(5, 0) };
106.
107.     Point polygon3[] = { new Point(0, 0), new Point(10, 0),
108.                          new Point(10, 10), new Point(0, 10), new Point(5, 5) };
109.
110.     System.out.println("Polygon 1 contains Polygon 2 :"
111.                         + DoesPolygonContainPolygon(polygon1, polygon2));
112.
113.     System.out.println("Polygon 3 contains Polygon 1 :"
114.                         + DoesPolygonContainPolygon(polygon3, polygon1));
115.
116. }
117. }
```

Output:

advertisement

```
$ javac Polygon_Containment_Test.java  
$ java Polygon_Containment_Test
```

```
Polygon 1 contains Polygon 2 :false  
Polygon 3 contains Polygon 1 :true
```

287. Java Program to Check if a Point d lies Inside or Outside a Circle Defined by Points a, b, c in a Plane

[« Prev](#)

[Next »](#)

This is a Java Program to check whether a point lies inside, outside or on the circle. For any point t (xt, yt) on the plane, its position with respect to the circle defined by 3 points (x1, y1), (x2, y2), (x3, y3).

$$s = (x - xt)^2 + (y - yt)^2 - r^2$$

If $s < 0$, t lies inside the circle; if $s > 0$, t lies outside the circle; if $s = 0$, t lies on the circle.

Here is the source code of the Java Program to Check if a Point d lies Inside or Outside a Circle Defined by Points a, b, c in a Plane. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether point d lies inside or outside the circle defined by a, b, c points
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Position_Point_WRT_Circle
6. {
7.     public static void main(String args[])
8.     {
9.         Random random = new Random();
10.
11.        int x1, y1, x2, y2, x3, y3;
12.        double m1, m2, c1, c2, r;
13.
14.        x1 = random.nextInt(10);
15.        y1 = random.nextInt(10);
16.        x2 = random.nextInt(10);
17.        y2 = random.nextInt(10);
18.        x3 = random.nextInt(10);
19.        y3 = random.nextInt(10);
20.
21.        m1 = (y1 - y2) / (x1 - x2);
22.        m2 = (y3 - y2) / (x3 - x2);
23.
24.        c1 = ((m1 * m2 * (y3 - y1)) + (m1 * (x2 + x3)) - (m2 * (x1 + x2)))
25.            / (2 * (m1 - m2));
26.        c2 = (((x1 + x2) / 2) - c1) / (-1 * m1)) + ((y1 + y2) / 2);
27.        r = Math.sqrt(((x3 - c1) * (x3 - c1)) + ((y3 - c2) * (y3 - c2)));
28.
29.        System.out.println("The points on the circle are: (" + x1 + ", " + y1
30.                            + "), (" + x2 + ", " + y2 + "), (" + x3 + ", " + y3 + ")");
31.        System.out.println("The center of the circle is (" + c1 + ", " + c2
32.                            + ") and radius is " + r);
33.
34.        System.out.println("Enter the point : <x>,<y>");
35.        Scanner scan = new Scanner(System.in);
36.        int x, y;
37.        x = scan.nextInt();
38.        y = scan.nextInt();
39.
40.        double s = ((x - c1) * (x - c1)) + ((y - c2) * (y - c1)) - (r * r);
41.        if (s < 0)
42.            System.out.println("The point lies inside the circle");
43.        else if (s > 0)
44.            System.out.println("The point lies outside the circle");
45.        else
46.            System.out.println("The point lies on the circle");
```

```
47.     scan.close();
48. }
49.}
```

Output:

advertisement

```
$ javac Position_Point_WRT_Circle.java
$ java Position_Point_WRT_Circle
```

The points on the circle are: (5, 9), (4, 7), (2, 0)
The center of the circle is (34.5, 23.25) and radius is 39.960136386153636
Enter the point : <x>,<y>
3 5
The point lies inside the circle

The points on the circle are: (0, 1), (2, 3), (5, 4)
The center of the circle is (3.5, 4.5) and radius is 1.5811388300841898
Enter the point : <x>,<y>
1 2
The point lies outside the circle

288. Java Program to Apply Above-Below-on Test to Find the Position of a Point with respect to a Line

[« Prev](#)

[Next »](#)

This is a Java Program to check whether a point lies below, above or on the line. For any point t (xt, yt) on the plane, its position with respect to the line L connecting p and q is found by calculating the scalar s:

$$s = A \cdot xt + B \cdot yt + C$$

If $s < 0$, t lies in the clockwise halfplane of L; if $s > 0$, t lies on the counter-clockwise halfplane; if $s = 0$, t lies on L.

For example, the equation of the line connecting points (2, 2) and (4, 5) is $-3x + 2y + 2 = 0$. The point (6, 3) lies in the clockwise halfplane of this line, because $(-3)(6) + (2)(3) + 2 = -10$. Conversely, the point (0, 5) lies in the other halfplane as $(-3)(0) + (2)(5) + 2 = 12$.

Here is the source code of the Java Program to Apply Above-Below-on Test to Find the Position of a Point with respect to a Line. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether a point lies on, above or below the gievn line
2. import java.util.Random;
3. import java.util.Scanner;
4.
5. public class Position_Point_WRT_Line
6. {
7.     public static void main(String args[])
8.     {
9.         Random random = new Random();
10.        int x1, x2, y1, y2;
11.        x1 = random.nextInt(10);
12.        x2 = random.nextInt(10);
13.        y1 = random.nextInt(10);
14.        y2 = random.nextInt(10);
15.
16.        System.out.println("The Equation of the line is : (" + (y2 - y1)
17.                           + ")x+" + (x1 - x2) + ")y+" + (x2 * y1 - x1 * y2) + " = 0");
18.
19.        System.out.println("Enter the point : <x>,<y>");
20.        Scanner scan = new Scanner(System.in);
21.        int x, y;
22.        x = scan.nextInt();
23.        y = scan.nextInt();
24.
25.        int s = (y2 - y1) * x + (x1 - x2) * y + (x2 * y1 - x1 * y2);
26.        if(s<0)
27.            System.out
28.                .println("The point lies below the line or left side of the line");
29.        else if (s>0)
30.            System.out
31.                .println("The point lies above the line or right side of the line");
32.        else
33.            System.out.println("The point lies on the line");
34.        scan.close();
35.    }
36.}
```

Output:

advertisement

```
$ javac Position_Point_WRT_Line.java  
$ java Position_Point_WRT_Line
```

The Equation of the line is : $(-2)x + (-9)y + (81) = 0$

Enter the point : <x>,<y>

2

3

The point lies above the line or right side of the line

289. Java Program to Check Whether a Given Point is in a Given Polygon

[« Prev](#)

[Next »](#)

This is a Java Program to check whether a point lies inside, outside or on the Polygon. Following is a simple idea to check whether a point is inside or outside.

- 1) Draw a horizontal line to the right of each point and extend it to infinity
- 2) Count the number of times the line intersects with polygon edges.
- 3) A point is inside the polygon if either count of intersections is odd or point lies on an edge of polygon. If none of the conditions is true, then point lies outside.

Here is the source code of the Java Program to Check Whether a Given Point is in a Given Polygon. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to check whether a point lies in a polygon or not
2. class Point
3. {
4.     int x, y;
5.
6.     Point()
7.     {}
8.
9.     Point(int p, int q)
10.    {
11.        x = p;
12.        y = q;
13.    }
14.}
15.
16. public class Position_Point_WRT_Polygon
17. {
18.
19.     public static boolean onSegment(Point p, Point q, Point r)
20.     {
21.         if (q.x <= Math.max(p.x, r.x) && q.x >= Math.min(p.x, r.x)
22.             && q.y <= Math.max(p.y, r.y) && q.y >= Math.min(p.y, r.y))
23.             return true;
24.         return false;
25.     }
26.
27.     public static int orientation(Point p, Point q, Point r)
28.     {
29.         int val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) * (r.y - q.y);
30.
31.         if (val == 0)
32.             return 0;
33.         return (val > 0) ? 1 : 2;
34.     }
35.
36.     public static boolean doIntersect(Point p1, Point q1, Point p2, Point q2)
37.     {
38.
39.         int o1 = orientation(p1, q1, p2);
40.         int o2 = orientation(p1, q1, q2);
41.         int o3 = orientation(p2, q2, p1);
42.         int o4 = orientation(p2, q2, q1);
43.
44.         if (o1 != o2 && o3 != o4)
45.             return true;
46.
47.         if (o1 == 0 && onSegment(p1, p2, q1))
```

```

48.     return true;
49.
50.    if (o2 == 0 && onSegment(p1, q2, q1))
51.        return true;
52.
53.    if (o3 == 0 && onSegment(p2, p1, q2))
54.        return true;
55.
56.    if (o4 == 0 && onSegment(p2, q1, q2))
57.        return true;
58.
59.    return false;
60. }
61.
62. public static boolean isInside(Point polygon[], int n, Point p)
63. {
64.     int INF = 10000;
65.     if (n < 3)
66.         return false;
67.
68.     Point extreme = new Point(INF, p.y);
69.
70.     int count = 0, i = 0;
71.     do
72.     {
73.         int next = (i + 1) % n;
74.         if (doIntersect(polygon[i], polygon[next], p, extreme))
75.         {
76.             if (orientation(polygon[i], p, polygon[next]) == 0)
77.                 return onSegment(polygon[i], p, polygon[next]);
78.
79.             count++;
80.         }
81.         i = next;
82.     } while (i != 0);
83.
84.     return (count & 1) == 1 ? true : false;
85. }
86.
87. public static void main(String args[])
88. {
89.     Point polygon1[] = { new Point(0, 0), new Point(10, 0),
90.                          new Point(10, 10), new Point(0, 10) };
91.     int n = 4;
92.
93.     Point p = new Point(20, 20);
94.     System.out.println("Point P(" + p.x + ", " + p.y
95.                        + ") lies inside polygon1: " + isInside(polygon1, n, p));
96.     p = new Point(5, 5);
97.     System.out.println("Point P(" + p.x + ", " + p.y
98.                        + ") lies inside polygon1: " + isInside(polygon1, n, p));
99.
100.    Point polygon2[] = { new Point(0, 0), new Point(5, 5), new Point(5, 0) };
101.    n = 3;
102.
103.    p = new Point(3, 3);
104.    System.out.println("Point P(" + p.x + ", " + p.y
105.                        + ") lies inside polygon2: " + isInside(polygon2, n, p));
106.    p = new Point(5, 1);
107.    System.out.println("Point P(" + p.x + ", " + p.y
108.                        + ") lies inside polygon2: " + isInside(polygon2, n, p));
109.    p = new Point(8, 1);
110.    System.out.println("Point P(" + p.x + ", " + p.y
111.                        + ") lies inside polygon2: " + isInside(polygon2, n, p));
112.
113.    Point polygon3[] = { new Point(0, 0), new Point(10, 0),

```

```
114.     new Point(10, 10), new Point(0, 10), new Point(5, 5) };
115.     n = 5;
116.
117.     p = new Point(-1, 10);
118.     System.out.println("Point P(" + p.x + ", " + p.y
119.             + ") lies inside polygon3: " + isInside(polygon3, n, p));
120. }
121.}
```

Output:

advertisement

```
$ javac Position_Point_WRT_Polygon.java
$ java Position_Point_WRT_Polygon
```

```
Point P(20, 20) lies inside polygon1: false
Point P(5, 5) lies inside polygon1: true
Point P(3, 3) lies inside polygon2: true
Point P(5, 1) lies inside polygon2: true
Point P(8, 1) lies inside polygon2: false
Point P(-1, 10) lies inside polygon3: false
```

290. Java Program to Implement Quick Hull Algorithm to Find Convex Hull

[« Prev](#)

[Next »](#)

This is a Java Program to implement Quick Hull Algorithm to find convex hull. Here we'll talk about the Quick Hull algorithm, which is one of the easiest to implement and has a reasonable expected running time of $O(n \log n)$.

Here is the source code of the Java Program to Implement Quick Hull Algorithm to Find Convex Hull. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find a points in convex hull using quick hull method
2. //source: Alexander Hrishov's website
3. //URL: http://www.ahristov.com/tutorial/geometry-games/convex-hull.html
4.
5. import java.util.ArrayList;
6. import java.util.Scanner;
7.
8. public class QuickHull
9. {
10.     public ArrayList<Point> quickHull(ArrayList<Point> points)
11.     {
12.         ArrayList<Point> convexHull = new ArrayList<Point>();
13.         if (points.size() < 3)
14.             return (ArrayList) points.clone();
15.
16.         int minPoint = -1, maxPoint = -1;
17.         int minX = Integer.MAX_VALUE;
18.         int maxX = Integer.MIN_VALUE;
19.         for (int i = 0; i < points.size(); i++)
20.         {
21.             if (points.get(i).x < minX)
22.             {
23.                 minX = points.get(i).x;
24.                 minPoint = i;
25.             }
26.             if (points.get(i).x > maxX)
27.             {
28.                 maxX = points.get(i).x;
29.                 maxPoint = i;
30.             }
31.         }
32.         Point A = points.get(minPoint);
33.         Point B = points.get(maxPoint);
34.         convexHull.add(A);
35.         convexHull.add(B);
36.         points.remove(A);
37.         points.remove(B);
38.
39.         ArrayList<Point> leftSet = new ArrayList<Point>();
40.         ArrayList<Point> rightSet = new ArrayList<Point>();
41.
42.         for (int i = 0; i < points.size(); i++)
43.         {
44.             Point p = points.get(i);
45.             if (pointLocation(A, B, p) == -1)
46.                 leftSet.add(p);
47.             else if (pointLocation(A, B, p) == 1)
48.                 rightSet.add(p);
49.         }
50.         hullSet(A, B, rightSet, convexHull);
51.         hullSet(B, A, leftSet, convexHull);
```

```

52.
53.     return convexHull;
54. }
55.
56. public int distance(Point A, Point B, Point C)
57. {
58.     int ABx = B.x - A.x;
59.     int ABy = B.y - A.y;
60.     int num = ABx * (A.y - C.y) - ABy * (A.x - C.x);
61.     if (num < 0)
62.         num = -num;
63.     return num;
64. }
65.
66. public void hullSet(Point A, Point B, ArrayList<Point> set,
67.                      ArrayList<Point> hull)
68. {
69.     int insertPosition = hull.indexOf(B);
70.     if (set.size() == 0)
71.         return;
72.     if (set.size() == 1)
73.     {
74.         Point p = set.get(0);
75.         set.remove(p);
76.         hull.add(insertPosition, p);
77.         return;
78.     }
79.     int dist = Integer.MIN_VALUE;
80.     int furthestPoint = -1;
81.     for (int i = 0; i < set.size(); i++)
82.     {
83.         Point p = set.get(i);
84.         int distance = distance(A, B, p);
85.         if (distance > dist)
86.         {
87.             dist = distance;
88.             furthestPoint = i;
89.         }
90.     }
91.     Point P = set.get(furthestPoint);
92.     set.remove(furthestPoint);
93.     hull.add(insertPosition, P);
94.
95. // Determine who's to the left of AP
96. ArrayList<Point> leftSetAP = new ArrayList<Point>();
97. for (int i = 0; i < set.size(); i++)
98. {
99.     Point M = set.get(i);
100.    if (pointLocation(A, P, M) == 1)
101.    {
102.        leftSetAP.add(M);
103.    }
104. }
105.
106. // Determine who's to the left of PB
107. ArrayList<Point> leftSetPB = new ArrayList<Point>();
108. for (int i = 0; i < set.size(); i++)
109. {
110.     Point M = set.get(i);
111.     if (pointLocation(P, B, M) == 1)
112.     {
113.         leftSetPB.add(M);
114.     }
115. }
116. hullSet(A, P, leftSetAP, hull);
117. hullSet(P, B, leftSetPB, hull);

```

```

118.
119. }
120.
121. public int pointLocation(Point A, Point B, Point P)
122. {
123.     int cp1 = (B.x - A.x) * (P.y - A.y) - (B.y - A.y) * (P.x - A.x);
124.     if (cp1 > 0)
125.         return 1;
126.     else if (cp1 == 0)
127.         return 0;
128.     else
129.         return -1;
130. }
131.
132. public static void main(String args[])
133. {
134.     System.out.println("Quick Hull Test");
135.     Scanner sc = new Scanner(System.in);
136.     System.out.println("Enter the number of points");
137.     int N = sc.nextInt();
138.
139.     ArrayList<Point> points = new ArrayList<Point>();
140.     System.out.println("Enter the coordinates of each points: <x> <y>");
141.     for (int i = 0; i < N; i++)
142.     {
143.         int x = sc.nextInt();
144.         int y = sc.nextInt();
145.         Point e = new Point(x, y);
146.         points.add(i, e);
147.     }
148.
149.     QuickHull qh = new QuickHull();
150.     ArrayList<Point> p = qh.quickHull(points);
151.     System.out
152.         .println("The points in the Convex hull using Quick Hull are: ");
153.     for (int i = 0; i < p.size(); i++)
154.         System.out.println("(" + p.get(i).x + ", " + p.get(i).y + ")");
155.     sc.close();
156. }
157. }

```

Output:

advertisement

```
$ javac QuickHull.java
$ java QuickHull
```

```
Quick Hull Test
Enter the number of points
4
Enter the coordinates of each points: <x> <y>
12 32
45 98
65 12
10 30
The points in the Convex hull using Quick Hull are:
(10, 30)
(45, 98)
(65, 12)
```

291. Java Program to Compute the Volume of a Tetrahedron Using Determinants

[« Prev](#)

[Next »](#)

This is a Java Program to compute volume of tetrahedron using determinants. Call the four vertices of the tetrahedron (a, b, c), (d, e, f), (g, h, i), and (p, q, r). Now create a 4-by-4 matrix in which the coordinate triples form the columns of the matrix, with a row of 1's appended at the bottom:

a	d	g	p
b	e	h	q
c	f	i	r
1	1	1	1
a-p		d-p	g-p
b-q		e-q	h-q
c-r f-r i-r			

The volume of the tetrahedron is 1/6 times the absolute value of the matrix determinant. For any 4-by-4 matrix that has a row of 1's along the bottom, you can compute the determinant with a simplification formula that reduces the problem

to

a

3-by-3

matrix

a-p
b-q
c-r f-r i-r

Here is the source code of the Java Program to Compute the Volume of a Tetrahedron Using Determinants. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to find the volume of tetrahedron using a method of determinant
2. import java.util.Random;
3.
4. public class Volume_Tetrahedron_Determinants
5. {
6.     public static double determinant(double A[][], int N)
7.     {
8.         double det = 0;
9.         if (N == 1)
10.        {
11.            det = A[0][0];
12.        } else if (N == 2)
13.        {
14.            det = A[0][0] * A[1][1] - A[1][0] * A[0][1];
15.        } else
16.        {
17.            det = 0;
18.            for (int j1 = 0; j1 < N; j1++)
19.            {
20.                double[][] m = new double[N - 1][];
21.                for (int k = 0; k < (N - 1); k++)
22.                {
23.                    m[k] = new double[N - 1];
24.                }
25.                for (int i = 1; i < N; i++)
26.                {
27.                    int j2 = 0;
28.                    for (int j = 0; j < N; j++)
29.                    {
30.                        if (j == j1)
31.                            continue;
32.                        m[i - 1][j2] = A[i][j];
33.                        j2++;
34.                    }
35.                }
36.                det += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
37.                  * determinant(m, N - 1);
38.            }

```

```

39.     }
40.     return det;
41. }
42.
43. public static void main(String args[])
44. {
45.     Random random = new Random();
46.     int x1, x2, x3, x4, y1, y2, y3, y4, z1, z2, z3, z4;
47.     x1 = random.nextInt(10);
48.     x2 = random.nextInt(10);
49.     x3 = random.nextInt(10);
50.     x4 = random.nextInt(10);
51.     y1 = random.nextInt(10);
52.     y2 = random.nextInt(10);
53.     y3 = random.nextInt(10);
54.     y4 = random.nextInt(10);
55.     z1 = random.nextInt(10);
56.     z2 = random.nextInt(10);
57.     z3 = random.nextInt(10);
58.     z4 = random.nextInt(10);
59.
60.     double[][] mat = new double[4][4];
61.     mat[0][0] = x1;
62.     mat[0][1] = x2;
63.     mat[0][2] = x3;
64.     mat[0][3] = x4;
65.     mat[1][0] = y1;
66.     mat[1][1] = y2;
67.     mat[1][2] = y3;
68.     mat[1][3] = y4;
69.     mat[2][0] = z1;
70.     mat[2][1] = z2;
71.     mat[2][2] = z3;
72.     mat[2][3] = z4;
73.     mat[3][0] = 1;
74.     mat[3][1] = 1;
75.     mat[3][2] = 1;
76.     mat[3][3] = 1;
77.
78.     System.out
79.         .println("The matrix formed by the coordinates of the tetrahedron is: ");
80.     for (int i = 0; i < 4; i++)
81.     {
82.         for (int j = 0; j < 4; j++)
83.             System.out.print(mat[i][j] + " ");
84.         System.out.println();
85.     }
86.     double[][] matrix = new double[3][3];
87.
88.     matrix[0][0] = x1 - x4;
89.     matrix[0][1] = x2 - x4;
90.     matrix[0][2] = x3 - x4;
91.     matrix[1][0] = y1 - y4;
92.     matrix[1][1] = y2 - y4;
93.     matrix[1][2] = y3 - y4;
94.     matrix[2][0] = z1 - z4;
95.     matrix[2][1] = z2 - z4;
96.     matrix[2][2] = z3 - z4;
97.     System.out.println("Matrix after simplification: ");
98.     for (int i = 0; i < 3; i++)
99.     {
100.         for (int j = 0; j < 3; j++)
101.             System.out.print(matrix[i][j] + " ");
102.         System.out.println();
103.     }
104.     double det = determinant(matrix, 3) / 6;

```

```

105. if (det < 0)
106.     System.out.println("The Area of the tetrahedron formed by (" + x1
107.             + "," + y1 + "," + z1 + "),(," + x2 + "," + y2 + "," + z2
108.             + "),(," + x3 + "," + y3 + "," + z3 + "), = " + (det * -1));
109. else
110.     System.out.println("The Area of the tetrahedron formed by (" + x1
111.             + "," + y1 + "," + z1 + "),(," + x2 + "," + y2 + "," + z2
112.             + "),(," + x3 + "," + y3 + "," + z3 + "), = " + (det * -1));
113. }
114.

```

Output:

advertisement

```
$ javac Volume_Tetrahedron_Determinants.java
$ java Volume_Tetrahedron_Determinants
```

The matrix formed by the coordinates of the tetrahedron is:

```
0.0 9.0 6.0 0.0
4.0 2.0 1.0 1.0
3.0 4.0 7.0 5.0
1.0 1.0 1.0 1.0
```

Matrix after simplification:

```
0.0 9.0 6.0
3.0 1.0 0.0
-2.0 -1.0 2.0
```

The Area of the tetrahedron formed by (0,4,3),(9,2,4),(6,1,7), = 10.0

292. Java Program to Delete a Particular Node in a Tree Without Using Recursion

[« Prev](#)

[Next »](#)

This is a Java Program to delete a particular node from the tree without using recursion.

Here is the source code of the Java Program to Delete a Particular Node in a Tree Without Using Recursion. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to delete a particular node from the tree without using a recursion
2. import java.util.Scanner;
3.
4. class BinaryNode
5. {
6.     private int    Key;
7.     private Object  Data;
8.     private BinaryNode Left;
9.     private BinaryNode Right;
10.
11.    public BinaryNode(int k, Object d)
12.    {
13.        Key = k;
14.        Data = d;
15.        Left = null;
16.        Right = null;
17.    }
18.
19.    // Get Operations
20.    public int gKey()
21.    {
22.        return Key;
23.    }
24.
25.    public Object gData()
26.    {
27.        return Data;
28.    }
29.
30.    public BinaryNode gLeft()
31.    {
32.        return Left;
33.    }
34.
35.    public BinaryNode gRight()
36.    {
37.        return Right;
38.    }
39.
40.    // Set Operations
41.    public void sKey(int AValue)
42.    {
43.        Key = AValue;
44.    }
45.
46.    public void sData(Object AValue)
47.    {
48.        Data = AValue;
49.    }
50.
51.    public void sLeft(BinaryNode AValue)
52.    {
53.        Left = AValue;
```

```
54. }
55.
56. public void sRight(BinaryNode AValue)
57. {
58.     Right = AValue;
59. }
60.}
61.
62.public class BTree
63.{
64.    private BinaryNode Root;
65.    private int NoOfNodes;
66.
67.    private BTree()
68.    { // constructor
69.        Root = null;
70.        NoOfNodes = 0;
71.    }
72.
73.    public boolean IsEmpty()
74.    {
75.        return (NoOfNodes == 0);
76.    }
77.
78.    public BinaryNode gRoot()
79.    {
80.        return Root;
81.    }
82.
83.    public int Count()
84.    {
85.        return NoOfNodes;
86.    }
87.
88.    public int Size(BinaryNode ATree)
89.    {
90.        if (ATree == null)
91.            return 0;
92.        else
93.            return (1 + Size(ATree.gLeft()) + Size(ATree.gRight())));
94.    }
95.
96.    public int Height(BinaryNode ATree)
97.    {
98.        if (ATree == null)
99.            return 0;
100.       else
101.           return (1 + Math.max(Height(ATree.gLeft()), Height(ATree.gRight())));
102.    }
103.
104.   public void PreOrder(BinaryNode ATree)
105.   {
106.       if (ATree != null)
107.       {
108.           System.out.print(ATree.gKey() + " ");
109.           PreOrder(ATree.gLeft());
110.           PreOrder(ATree.gRight());
111.       }
112.   }
113.
114.   public void InOrder(BinaryNode ATree)
115.   {
116.       if (ATree != null)
117.       {
118.           InOrder(ATree.gLeft());
119.           System.out.print(ATree.gKey() + " ");
```

```

120.     InOrder(ATree.gRight());
121. }
122. }
123.
124. public void PostOrder(BinaryNode ATree)
125. {
126.     if (ATree != null)
127.     {
128.         PostOrder(ATree.gLeft());
129.         PostOrder(ATree.gRight());
130.         System.out.print(ATree.gKey() + " ");
131.     }
132. }
133.
134. public void Insert(int AId, Object AValue)
135. {
136.     BinaryNode Temp, Current, Parent;
137.     if (Root == null)
138.     {
139.         Temp = new BinaryNode(AId, AValue);
140.         Root = Temp;
141.         NoOfNodes++;
142.     } else
143.     {
144.         Temp = new BinaryNode(AId, AValue);
145.         Current = Root;
146.         while (true)
147.         {
148.             Parent = Current;
149.             if (AId < Current.gKey())
150.             {
151.                 Current = Current.gLeft();
152.                 if (Current == null)
153.                 {
154.                     Parent.sLeft(Temp);
155.                     NoOfNodes++;
156.                     return;
157.                 }
158.             } else
159.             {
160.                 Current = Current.gRight();
161.                 if (Current == null)
162.                 {
163.                     Parent.sRight(Temp);
164.                     NoOfNodes++;
165.                     return;
166.                 }
167.             }
168.         }
169.     }
170. }
171. }
172.
173. public BinaryNode Find(int AKey)
174. {
175.     BinaryNode Current = null;
176.     if (!IsEmpty())
177.     {
178.         Current = Root; // start search at top of tree
179.         while (Current.gKey() != AKey)
180.         {
181.             if (AKey < Current.gKey())
182.                 Current = Current.gLeft();
183.             else
184.                 Current = Current.gRight();
185.             if (Current == null)

```

```

186.         return null;
187.     }
188. }
189. return Current;
190. }
191.
192. public BinaryNode GetSuccessor(BinaryNode ANode)
193. {
194.     BinaryNode Current, Successor, SuccessorParent;
195.     Successor = ANode;
196.     SuccessorParent = ANode;
197.     Current = ANode.gRight();
198.     while (Current != null)
199.     {
200.         SuccessorParent = Successor;
201.         Successor = Current;
202.         Current = Current.gLeft();
203.     }
204.     if (Successor != ANode.gRight())
205.     {
206.         SuccessorParent.sLeft(Successor.gRight());
207.         Successor.sRight(ANode.gRight());
208.     }
209.     return Successor;
210. }
211.
212. public boolean Delete(int AKey)
213. {
214.     BinaryNode Current, Parent;
215.     boolean IsLeftChild = true;
216.     Current = Root;
217.     Parent = Root;
218.     while (Current.gKey() != AKey)
219.     {
220.         Parent = Current;
221.         if (AKey < Current.gKey())
222.         {
223.             IsLeftChild = true;
224.             Current = Current.gLeft();
225.         } else
226.         {
227.             IsLeftChild = false;
228.             Current = Current.gRight();
229.         }
230.         if (Current == null)
231.             return false;
232.     }
233.
234.     if (Current.gLeft() == null && Current.gRight() == null)
235.     {
236.         if (Current == Root)
237.             Root = Current.gLeft();
238.         else if (IsLeftChild)
239.             Parent.sLeft(Current.gRight());
240.         else
241.             Parent.sRight(Current.gRight());
242.     }
243.
244.     else
245.     {
246.         if (Current.gRight() == null)
247.         {
248.             if (Current == Root)
249.                 Root = Current.gRight();
250.             else if (IsLeftChild)
251.                 Parent.sLeft(Current.gLeft());

```

```

252.         else
253.             Parent.sRight(Current.gLeft());
254.     }
255.
256.     else
257.     {
258.         if (Current.gLeft() == null)
259.         {
260.             if (Current == Root)
261.                 Root = Current.gLeft();
262.             else if (IsLeftChild)
263.                 Parent.sLeft(Current.gRight());
264.             else
265.                 Parent.sRight(Current.gRight());
266.         }
267.
268.         else
269.         {
270.             BinaryNode Successor = GetSuccessor(Current);
271.             if (Current == Root)
272.                 Root = Successor;
273.             else if (IsLeftChild)
274.                 Parent.sLeft(Successor);
275.             else
276.                 Parent.sRight(Successor);
277.             Successor.sLeft(Current.gLeft());
278.         }
279.     }
280. }
281. NoOfNodes--;
282. return true;
283. }
284.
285. public static void main(String[] Args)
286. {
287.     BTree MyTree = new BTree();
288.     BinaryNode NodeAt;
289.
290.     System.out.println("Enter the elements in the tree");
291.     int N = 5;
292.     Scanner sc = new Scanner(System.in);
293.     for (int i = 0; i < N; i++)
294.         MyTree.Insert(sc.nextInt(), i);
295.
296.     System.out.print("\nInorder : ");
297.     MyTree.InOrder(MyTree.gRoot());
298.     System.out.print("\nPreorder : ");
299.     MyTree.PreOrder(MyTree.gRoot());
300.     System.out.print("\nPostorder : ");
301.     MyTree.PostOrder(MyTree.gRoot());
302.
303.     System.out.println("\nEnter the element to be deleted: ");
304.     int delete = sc.nextInt();
305.
306.     MyTree.Delete(delete);
307.
308.     System.out.print("\nInorder : ");
309.     MyTree.InOrder(MyTree.gRoot());
310.     System.out.print("\nPreorder : ");
311.     MyTree.PreOrder(MyTree.gRoot());
312.     System.out.print("\nPostorder : ");
313.     MyTree.PostOrder(MyTree.gRoot());
314.
315.     sc.close();
316. }
317.

```

Output:

advertisement

```
$ javac BTree.java  
$ java BTree
```

Enter the elements in the tree
54 12 32 19 45

Inorder : 12 19 32 45 54
Preorder : 54 12 32 19 45
Postorder : 19 45 32 12 54

Enter the element to be deleted:
12

Inorder : 19 32 45 54
Preorder : 54 32 19 45
Postorder : 19 45 32 54

293. Java Program to Compute Discrete Fourier Transform Using Naive Approach

[« Prev](#)

[Next »](#)

This is the java implementation of Naive DFT approach over function. Formula for calculating the coefficient is $X(k) = \text{Sum}(x(n)*\cos(2*\pi*k*n/N) - i\text{Sum}(x(n)*\sin(2*\pi*k*n/N))$ over 0 to N-1. This approach tries to many transforms using different values of k from 0 to N-1.

Here is the source code of the Java Program to Compute Discrete Fourier Transform Using Naive Approach. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. //This is a java program to perform the DFT using naive approach
2. import java.util.Scanner;
3.
4. public class DFT_Naive_Approach
5. {
6.     double real, img;
7.     public DFT_Naive_Approach()
8.     {
9.         this.real = 0.0;
10.        this.img = 0.0;
11.    }
12.    public static void main(String args[])
13.    {
14.        int N = 10;
15.        Scanner sc = new Scanner(System.in);
16.        System.out.println("Discrete Fourier Transform using naive method");
17.        System.out.println("Enter the coefficient of simple linear function:");
18.        System.out.println("ax + by = c");
19.        double a = sc.nextDouble();
20.        double b = sc.nextDouble();
21.        double c = sc.nextDouble();
22.
23.        double []function = new double[N];
24.        for(int i=0; i<N; i++)
25.        {
26.            function[i] = ((a*(double)i) + (b*(double)i)) - c;
27.        }
28.
29.        System.out.println("Enter the max K value: ");
30.        int k = sc.nextInt();
31.
32.        DFT_Naive_Approach []dft_val = new DFT_Naive_Approach[k];
33.
34.        System.out.println("The coefficients are: ");
35.        for(int j=0; j<k; j++)
36.        {
37.            dft_val[j] = new DFT_Naive_Approach();
38.            for(int i=0; i<N; i++)
39.            {
40.                dft_val[j].real += function[i] * Math.cos((2 * i * j * Math.PI) / N);
41.                dft_val[j].img += function[i] * Math.sin((2 * i * j * Math.PI) / N);
42.            }
43.            System.out.println("(" + dft_val[j].real + " - " + "(" + dft_val[j].img + " i)");
44.        }
45.        sc.close();
46.    }
47. }
```

Output:

```
$ javac DFT_Naive_Approach.java  
$ java DFT_Naive_Approach
```

Discrete Fourier Transform using naive method

Enter the coefficient of simple linear funtion:

$ax + by = c$

1 2 3

Enter the max K value:

20

The coefficients are:

```
(105.0) - (0.0 i)  
(-15.00000000000001) - (-46.1652530576288 i)  
(-15.00000000000001) - (-20.6457288070676 i)  
(-15.00000000000005) - (-10.898137920080407 i)  
(-15.00000000000004) - (-4.873795443493586 i)  
(-15.0) - (1.4695761589768243E-14 i)  
(-14.999999999999996) - (4.873795443493611 i)  
(-15.000000000000103) - (10.898137920080355 i)  
(-14.999999999999968) - (20.64572880706762 i)  
(-14.999999999999922) - (46.16525305762871 i)  
(105.0) - (-1.7634913907721884E-13 i)  
(-15.00000000000012) - (-46.16525305762882 i)  
(-15.000000000000053) - (-20.645728807067577 i)  
(-14.999999999999911) - (-10.898137920080416 i)  
(-15.000000000000037) - (-4.87379544349373 i)  
(-15.0) - (1.0803613098771371E-13 i)  
(-14.99999999999984) - (4.873795443493645 i)  
(-14.9999999999996) - (10.89813792008029 i)  
(-14.999999999999677) - (20.645728807067492 i)  
(-14.99999999999769) - (46.16525305762875 i)
```

294. Java Program to Perform Arithmetic Operations on Numbers of Size Greater than that of Int Without Using any Data Type of Size Greater than Int

[« Prev](#)

[Next »](#)

This is a java program to perform arithmetic operations on numbers which are greater than size of the integers.

Here is the source code of the Java Program to Perform Arithmetic Operations on Numbers of Size Greater than that of Int Without Using any Data Type of Size Greater than Int. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. package com.sanfoundry.numerical;
2.
3. import java.util.Scanner;
4.
5. public class BigNumber
6. {
7.     static final int MAXDIGITS = 100; /* maximum length bignum */
8.     static final int PLUS    = 1; /* positive sign bit */
9.     static final int MINUS   = -1; /* negative sign bit */
10.    char      digits[]; /* represent the number */
11.    int       signbit; /* 1 if positive, -1 if negative */
12.    int       lastdigit; /* index of high-order digit */
13.
14.    BigNumber()
15.    {
16.        digits = new char[MAXDIGITS];
17.        intToBigNumber(0);
18.    }
19.
20.    void printBigNumber()
21.    {
22.        int i;
23.        if (signbit == MINUS)
24.            System.out.printf("- ");
25.        for (i = lastdigit; i >= 0; i--)
26.            System.out.printf("%c", '0' + digits[i]);
27.        System.out.printf("\n");
28.    }
29.
30.    void intToBigNumber(int s)
31.    {
32.        if (s >= 0)
33.            signbit = PLUS;
34.        else
35.            signbit = MINUS;
36.        for (int i = 0; i < MAXDIGITS; i++)
37.            digits[i] = (char) 0;
38.        lastdigit = -1;
39.        int t = Math.abs(s);
40.        while (t > 0)
41.        {
42.            lastdigit++;
43.            digits[lastdigit] = (char) (t % 10);
44.            t /= 10;
45.        }
46.        if (s == 0)
47.            lastdigit = 0;
```

```

48. }
49.
50. BigNumber addBigNumber(BigNumber b)
51. {
52.     int carry, i;
53.     BigNumber c = new BigNumber();
54.     if (signbit == b.signbit)
55.         c.signbit = signbit;
56.     else
57.     {
58.         if (signbit == MINUS)
59.         {
60.             signbit = PLUS;
61.             c = b.subtractBigNumber(this);
62.             signbit = MINUS;
63.         }
64.         else
65.         {
66.             b.signbit = PLUS;
67.             c = this.subtractBigNumber(b);
68.             b.signbit = MINUS;
69.         }
70.     return c;
71. }
72.     c.lastdigit = Math.max(lastdigit, b.lastdigit) + 1;
73.     carry = 0;
74.     for (i = 0; i <= c.lastdigit; i++)
75.     {
76.         c.digits[i] = (char) ((carry + digits[i] + b.digits[i]) % 10);
77.         carry = (carry + digits[i] + b.digits[i]) / 10;
78.     }
79.     c.zeroJustify();
80.     return c;
81. }
82.
83. BigNumber subtractBigNumber(BigNumber b)
84. {
85.     int borrow, v, i;
86.     BigNumber c = new BigNumber();
87.     if (signbit == MINUS || b.signbit == MINUS)
88.     {
89.         b.signbit = -b.signbit;
90.         c = addBigNumber(b);
91.         b.signbit = -b.signbit;
92.     return c;
93. }
94.     if (compareBigNumber(b) == PLUS)
95.     {
96.         c = b.subtractBigNumber(this);
97.         c.signbit = MINUS;
98.     return c;
99. }
100.    c.lastdigit = Math.max(lastdigit, b.lastdigit);
101.    borrow = 0;
102.    for (i = 0; i <= c.lastdigit; i++)
103.    {
104.        v = digits[i] - borrow - b.digits[i];
105.        if (digits[i] > 0)
106.            borrow = 0;
107.        if (v < 0)
108.        {
109.            v = v + 10;
110.            borrow = 1;
111.        }
112.        c.digits[i] = (char) (v % 10);
113.    }

```

```

114.     c.zeroJustify();
115.     return c;
116. }
117.
118. int compareBigNumber(BigNumber b)
119. {
120.     int i;
121.     if (signbit == MINUS && b.signbit == PLUS)
122.         return PLUS;
123.     if (signbit == PLUS && b.signbit == MINUS)
124.         return MINUS;
125.     if (b.lastdigit > lastdigit)
126.         return PLUS * signbit;
127.     if (lastdigit > b.lastdigit)
128.         return MINUS * signbit;
129.     for (i = lastdigit; i >= 0; i--)
130.     {
131.         if (digits[i] > b.digits[i])
132.             return MINUS * signbit;
133.         if (b.digits[i] > digits[i])
134.             return PLUS * signbit;
135.     }
136.     return 0;
137. }
138.
139. void zeroJustify()
140. {
141.     while (lastdigit > 0 && digits[lastdigit] == 0)
142.         lastdigit--;
143.     if (lastdigit == 0 && digits[0] == 0)
144.         signbit = PLUS; /* hack to avoid -0 */
145. }
146.
147. void digitShift(int d)
148. {
149.     int i;
150.     if (lastdigit == 0 && digits[0] == 0)
151.         return;
152.     for (i = lastdigit; i >= 0; i--)
153.         digits[i + d] = digits[i];
154.     for (i = 0; i < d; i++)
155.         digits[i] = 0;
156.     lastdigit += d;
157. }
158.
159. BigNumber multiplyBigNumber(BigNumber b)
160. {
161.     BigNumber row = new BigNumber();
162.     BigNumber tmp = new BigNumber();
163.     BigNumber c = new BigNumber();
164.     int i, j;
165.     row.signbit = this.signbit;
166.     row.lastdigit = this.lastdigit;
167.     System.arraycopy(this.digits, 0, row.digits, 0, this.digits.length);
168.     for (i = 0; i <= b.lastdigit; i++)
169.     {
170.         for (j = 1; j <= b.digits[i]; j++)
171.         {
172.             tmp = c.addBigNumber(row);
173.             c = tmp;
174.         }
175.         row.digitShift(1);
176.     }
177.     c.signbit = signbit * b.signbit;
178.     c.zeroJustify();
179.     return c;

```

```

180. }
181.
182. BigNumber divideBigNumber(BigNumber b)
183. {
184.     BigNumber row = new BigNumber();
185.     BigNumber tmp = new BigNumber();
186.     BigNumber c = new BigNumber();
187.     int asign, bsign, i;
188.     c.signbit = signbit * b.signbit;
189.     asign = signbit;
190.     bsign = b.signbit;
191.     signbit = PLUS;
192.     b.signbit = PLUS;
193.     c.lastdigit = lastdigit;
194.     for (i = lastdigit; i >= 0; i--)
195.     {
196.         row.digitShift(1);
197.         row.digits[0] = digits[i];
198.         c.digits[i] = 0;
199.         while (row.compareBigNumber(b) != PLUS)
200.         {
201.             c.digits[i]++;
202.             tmp = row.subtractBigNumber(b);
203.             row = tmp;
204.         }
205.     }
206.     c.zeroJustify();
207.     signbit = asign;
208.     b.signbit = bsign;
209.     return c;
210. }
211. }
212.
213. public class ArithmeticOpsBigNumbers
214. {
215.     public static void main(String[] args)
216.     {
217.         int a, b;
218.         BigNumber n1 = new BigNumber();
219.         BigNumber n2 = new BigNumber();
220.         BigNumber n3 = new BigNumber();
221.         BigNumber zero = new BigNumber();
222.         Scanner sc = new Scanner(System.in);
223.         while (sc.hasNextInt())
224.         {
225.             a = sc.nextInt();
226.             b = sc.nextInt();
227.             System.out.printf("a = %d  b = %d\n", a, b);
228.             n1.intToBigNumber(a);
229.             n2.intToBigNumber(b);
230.             n3 = n1.addBigNumber(n2);
231.             System.out.print("Addition: ");
232.             n3.printBigNumber();
233.             System.out.print("Comparing Numbers a ? b: %d\n",
234.                             n1.compareBigNumber(n2));
235.             n3 = n1.subtractBigNumber(n2);
236.             System.out.print("Subtraction: ");
237.             n3.printBigNumber();
238.             n3 = n1.multiplyBigNumber(n2);
239.             System.out.print("Multiplication: ");
240.             n3.printBigNumber();
241.             zero.intToBigNumber(0);
242.             if (zero.compareBigNumber(n2) == 0)
243.                 System.out.printf("Division: NaN \n");
244.             else
245.             {

```

```
246.         n3 = n1.divideBigNumber(n2);
247.         System.out.printf("Division: ");
248.         n3.printBigNumber();
249.     }
250. }
251. sc.close();
252. }
253. }
```

Output:

advertisement

```
$ javac ArithmeticOpsBigNumbers.java
$ java ArithmeticOpsBigNumbers
```

```
12342424
12313423
a = 12342424 b = 12313423
Addition: 24655847
Comparing Numbers a ? b: -1
Subtraction: 29001
Multiplication: 151977487557352
Division: 1
```

295. Java Program to Perform Matrix Multiplication

[« Prev](#)

[Next »](#)

This is a java program to perform a simple matrix multiplication. For matrix multiplication to happen the column of the first matrix should be equal to the row of the second matrix.

Here is the source code of the Java Program to Perform Matrix Multiplication. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. // This is sample program for matrix multiplication
2. // The complexity of the algorithm is O(n^3)
3. package com.sanfoundry.numerical;
4.
5. import java.util.Scanner;
6.
7. public class MatixMultiplication
8. {
9.     public static void main(String args[])
10.    {
11.        int n;
12.        Scanner input = new Scanner(System.in);
13.        System.out.println("Enter the base of squared matrices");
14.        n = input.nextInt();
15.        int[][] a = new int[n][n];
16.        int[][] b = new int[n][n];
17.        int[][] c = new int[n][n];
18.        System.out.println("Enter the elements of 1st martix row wise \n");
19.        for (int i = 0; i < n; i++)
20.        {
21.            for (int j = 0; j < n; j++)
22.            {
23.                a[i][j] = input.nextInt();
24.            }
25.        }
26.        System.out.println("Enter the elements of 2nd martix row wise \n");
27.        for (int i = 0; i < n; i++)
28.        {
29.            for (int j = 0; j < n; j++)
30.            {
31.                b[i][j] = input.nextInt();
32.            }
33.        }
34.        System.out.println("Multiplying the matrices...");
35.        for (int i = 0; i < n; i++)
36.        {
37.            for (int j = 0; j < n; j++)
38.            {
39.                for (int k = 0; k < n; k++)
40.                {
41.                    c[i][j] = c[i][j] + a[i][k] * b[k][j];
42.                }
43.            }
44.        }
45.        System.out.println("The product is:");
46.        for (int i = 0; i < n; i++)
47.        {
48.            for (int j = 0; j < n; j++)
49.            {
50.                System.out.print(c[i][j] + " ");
51.            }
52.        }
53.    }
54. }
```

```
52.     System.out.println();
53. }
54.     input.close();
55. }
56. }
```

Output:

advertisement

Output:

```
$ javac MatixMultiplication.java
$ java MatixMultiplication
```

Enter the base of squared matrices:

3

Enter the elements of 1st martix row wise:

1 2 3

4 5 6

7 8 9

Enter the elements of 2nd martix row wise:

2 3 4

5 6 7

8 9 1

Multiplying the matrices...

The product is:

36 42 21

81 96 57

126 150 93

296. Java Program to Perform Optimal Paranthesization Using Dynamic Programming

[« Prev](#)

[Next »](#)

This is a java program to perform optimal paranthesization by making use of dymanic programming. This program determines the order in which the chain of matrices should be multiplied.

Here is the source code of the Java Program to Perform Optimal Paranthesization Using Dynamic Programming. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. package com.sanfoundry.numerical;
2.
3. import java.util.Scanner;
4.
5. public class OptimalParanthesizationUsingDP
6. {
7.     private int[][] m;
8.     private int[][] s;
9.     private int n;
10.
11.    public OptimalParanthesizationUsingDP(int[] p)
12.    {
13.        n = p.length - 1; // how many matrices are in the chain
14.        m = new int[n + 1][n + 1]; // overallocate m, so that we don't use index
15.                                // 0
16.        s = new int[n + 1][n + 1]; // same for s
17.        matrixChainOrder(p); // run the dynamic-programming algorithm
18.    }
19.
20.    private void matrixChainOrder(int[] p)
21.    {
22.        // Initial the cost for the empty subproblems.
23.        for (int i = 1; i <= n; i++)
24.            m[i][i] = 0;
25.        // Solve for chains of increasing length l.
26.        for (int l = 2; l <= n; l++)
27.        {
28.            for (int i = 1; i <= n - l + 1; i++)
29.            {
30.                int j = i + l - 1;
31.                m[i][j] = Integer.MAX_VALUE;
32.                // Check each possible split to see if it's better
33.                // than all seen so far.
34.                for (int k = i; k < j; k++)
35.                {
36.                    int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
37.                    if (q < m[i][j])
38.                    {
39.                        // q is the best split for this subproblem so far.
40.                        m[i][j] = q;
41.                        s[i][j] = k;
42.                    }
43.                }
44.            }
45.        }
46.    }
47.
48.    private String printOptimalParens(int i, int j)
49.    {
50.        if (i == j)
```

```

51.     return "A[" + i + "]";
52.   else
53.     return "(" + printOptimalParenthesis(i, s[i][j])
54.           + printOptimalParenthesis(s[i][j] + 1, j) + ")";
55. }
56.
57. public String toString()
58. {
59.   return printOptimalParenthesis(1, n);
60. }
61.
62. public static void main(String[] args)
63. {
64.   Scanner sc = new Scanner(System.in);
65.   System.out
66.       .println("Enter the array p[], which represents the chain of matrices such that the ith matrix Ai is of
67. dimension p[i-1] x p[i]");
68.   System.out.println("Enter the total length: ");
69.   int n = sc.nextInt();
70.   int arr[] = new int[n];
71.   System.out.println("Enter the dimensions: ");
72.   for (int i = 0; i < n; i++)
73.     arr[i] = sc.nextInt();
74.   OptimalParanthesizationUsingDP opudp = new OptimalParanthesizationUsingDP(
75.     arr);
76.   System.out.println("Matrices are of order: ");
77.   for (int i = 1; i < arr.length; i++)
78.   {
79.     System.out.println("A" + i + "-->" + arr[i - 1] + "x" + arr[i]);
80.   }
81.   System.out.println(opudp.toString());
82.   sc.close();
83. }

```

Output:

advertisement

```
$ javac OptimalParanthesizationUsingDP.java
$ java OptimalParanthesizationUsingDP
```

Enter the array p[], which represents the chain of matrices such that the ith matrix Ai is of dimension p[i-1] x p[i]
Enter the total length:

5
Enter the dimensions:

2 4 5 2 1

Matrices are of order:

A1-->2x4
A2-->4x5
A3-->5x2
A4-->2x1
(A[1](A[2](A[3]A[4])))

297. Java Program to Implement the Alexander Bogomolny's UnOrdered Permutation Algorithm for Elements From 1 to N

[« Prev](#)

[Next »](#)

This is a java program to implement Alexander Bogomolny's permutation algorithm. This version of program computes all possible permutations of numbers from 1 to N using Alexander Bogomolyn's algorithm.

Here is the source code of the Java Program to Implement the Alexander Bogomolny's UnOrdered Permutation Algorithm for Elements From 1 to N. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.combinatorial;
3. 
4. import java.util.Scanner;
5. 
6. public class AlexanderBogomolnyPermutation
7. {
8.     static int level = -1;
9. 
10.    public static void print(int[] value, int n)
11.    {
12.        if (value.length != 0)
13.        {
14.            for (int i = 0; i < value.length; i++)
15.            {
16.                System.out.print(value[i] + " ");
17.            }
18.            System.out.println();
19.        }
20.    }
21. 
22.    public static void visit(int[] Value, int N, int k)
23.    {
24.        level = level + 1;
25.        Value[k] = level;
26.        if (level == N)
27.            print(Value, N);
28.        else
29.            for (int i = 0; i < N; i++)
30.                if (Value[i] == 0)
31.                    visit(Value, N, i);
32.        level = level - 1;
33.        Value[k] = 0;
34.    }
35. 
36.    public static void main(String[] args)
37.    {
38.        Scanner sc = new Scanner(System.in);
39.        System.out.println("Enter the size of the sequence:");
40.        int n = sc.nextInt();
41.        int sequence[] = new int[n];
42.        for (int i = 0; i < n; i++)
43.        {
44.            sequence[i] = 0;
45.        }
46.        System.out.println("The permutations are: ");
47.        visit(sequence, n, 0);
48.        sc.close();
```

```
49. }  
50. }
```

Output:

advertisement

```
$ javac AlexanderBogomolnyPermutation.java  
$ java AlexanderBogomolnyPermutation
```

Enter the size of the sequence:

4

The permutations are:

```
1 2 3 4  
1 2 4 3  
1 3 2 4  
1 4 2 3  
1 3 4 2  
1 4 3 2  
2 1 3 4  
2 1 4 3  
3 1 2 4  
4 1 2 3  
3 1 4 2  
4 1 3 2  
2 3 1 4  
2 4 1 3  
3 2 1 4  
4 2 1 3  
3 4 1 2  
4 3 1 2  
2 3 4 1  
2 4 3 1  
3 2 4 1  
4 2 3 1  
3 4 2 1  
4 3 2 1
```

298. Java Program to Implement a Binary Search Algorithm for a Specific Search Sequence

[« Prev](#)

[Next »](#)

This is a java program to search sequence using binary search. This is a simple extension of binary search algorithm to find an element.

Here is the source code of the Java Program to Implement a Binary Search Algorithm for a Specific Search Sequence. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.combinatorial;
3. 
4. import java.util.Random;
5. import java.util.Scanner;
6. 
7. public class BinarySearchSequence
8. {
9.     public static void searchSequence(int[] array, int[] search)
10.    {
11.        int first, last, middle;
12.        first = 0;
13.        last = array.length - 1;
14.        boolean flag = true;
15.        for (int i = 0; i < search.length; i++)
16.        {
17.            middle = (first + last) / 2;
18.            while (first <= last && flag == true)
19.            {
20.                if (array[middle] < search[i])
21.                {
22.                    first = middle + 1;
23.                }
24.                else if (array[middle] == search[i])
25.                {
26.                    System.out.println(search[i] + " found at location "
27.                        + (middle + 1) + ".");
28.                    first = 0;
29.                    last = array.length - 1;
30.                    break;
31.                }
32.                else
33.                {
34.                    last = middle - 1;
35.                }
36.                middle = (first + last) / 2;
37.            }
38.            if (first > last)
39.            {
40.                System.out
41.                    .println(search[i] + " is not present in the list.");
42.                flag = false;
43.            }
44.        }
45.    }
46. 
47. public static void main(String args[])
48. {
49.     int c, n, search[], array[];
50.     Scanner in = new Scanner(System.in);
```

```
51. System.out.println("Enter number of elements: ");
52. n = in.nextInt();
53. array = new int[n];
54. Random rand = new Random();
55. for (c = 0; c < n; c++)
56. {
57.     array[c] = rand.nextInt(100);
58. }
59. System.out.println("Elements: ");
60. for (int i = 0; i < array.length; i++)
61. {
62.     System.out.print(array[i] + " ");
63. }
64. System.out.println("\nEnter length of sequence to find: ");
65. int m = in.nextInt();
66. search = new int[m];
67. System.out.println("Enter the sequence to find: ");
68. for (int i = 0; i < m; i++)
69. {
70.     search[i] = in.nextInt();
71. }
72. searchSequence(array, search);
73. in.close();
74. }
75. }
```

Output:

advertisement

```
$ javac BinarySearchSequence.java
$ java BinarySearchSequence
```

Enter number of elements:

10

Elements:

68 45 85 63 7 48 44 93 10 20

Enter length of sequence to find:

2

Enter the sequence to find:

7 48

7 found at location 5.

48 found at location 6.

Enter number of elements:

10

Elements:

60 52 44 55 55 25 34 97 24 18

Enter length of sequence to find:

3

Enter the sequence to find:

2 3 4

2 is not present in the list.

3 is not present in the list.

4 is not present in the list.

299. Java Program to Check if any Graph is Possible to be Constructed for a Given Degree Sequence

[« Prev](#)

[Next »](#)

This is a java program to check graph construction is possible or not using given degree sequence. If the sum of degree is even graph construction is possible, not otherwise.

Here is the source code of the Java Program to Check if any Graph is Possible to be Constructed for a Given Degree Sequence. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.combinatorial;
3.
4. import java.util.ArrayList;
5. import java.util.List;
6. import java.util.Scanner;
7.
8. public class CheckGraphConstuction
9. {
10.     public static Integer sum(List<Integer> list)
11.     {
12.         Integer sum = 0;
13.         for (Integer integer : list)
14.         {
15.             sum += integer;
16.         }
17.         return sum;
18.     }
19.
20.     public static void main(String[] args)
21.     {
22.         Scanner sc = new Scanner(System.in);
23.         System.out.println("Enter the number of vertices: ");
24.         Integer n = sc.nextInt();
25.         System.out
26.             .println("Enter the Degree Sequence: <Degree sequence is always in non-increasing order>");
27.         List<Integer> sequence = new ArrayList<Integer>();
28.         while (n > 0)
29.         {
30.             sequence.add(sc.nextInt());
31.             n--;
32.         }
33.         System.out.println(sequence.toString());
34.         if (sum(sequence) % 2 == 0)
35.         {
36.             System.out
37.                 .println("Graph can be constructed using the given sequence G=" +
38.                     + sequence.size() +
39.                     + ", " +
40.                     + (sum(sequence) / 2) +
41.                     + ")");
42.         }
43.         sc.close();
44.     }
45. }
```

Output:

```
$ javac CheckGraphConstuction.java  
$ java CheckGraphConstuction
```

Enter the number of vertices:

7

Enter the Degree Sequence: <Degree sequence is always in non-increasing order>

5 3 3 2 2 1 0

[5, 3, 3, 2, 2, 1, 0]

Graph can be constructed using the given sequence G=(7, 8).

Enter the number of vertices:

3

Enter the Degree Sequence: <Degree sequence is always in non-increasing order>

3 3 1

[3, 3, 1]

no soultion exists.

300. Java Program to Generate a Graph for a Given Fixed Degree Sequence

[« Prev](#)

[Next »](#)

This is a java program to generate a graph from given degree sequence. The degree sequence of an undirected graph is the non-increasing sequence of its vertex degrees. The degree sequence problem is the problem of finding some or all graphs with the degree sequence being a given non-increasing sequence of positive integers. A sequence which is the degree sequence of some graph, i.e. for which the degree sequence problem has a solution, is called a graphic or graphical sequence. As a consequence of the degree sum formula, any sequence with an odd sum, such as (3, 3, 1), cannot be realized as the degree sequence of a graph. The converse is also true: if a sequence has an even sum, it is the degree sequence of a multigraph. The construction of such a graph is straightforward: connect vertices with odd degrees in pairs by a matching, and fill out the remaining even degree counts by self-loops.

Here is the source code of the Java Program to Generate a Graph for a Given Fixed Degree Sequence. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.hinguapps.combinatorial;
3.
4. import java.util.ArrayList;
5. import java.util.List;
6. import java.util.Scanner;
7.
8. public class GraphUsingDegreeSequence
9. {
10.     Integer[][] adjacencyMatrix;
11.     List<Integer> degreeSequence;
12.
13.     private void addEdges(Integer v, Integer e)
14.     {
15.         for (int i = 0; i < adjacencyMatrix.length && e > 0; i++)
16.         {
17.             if (degreeSequence.get(i) != 0)
18.             {
19.                 adjacencyMatrix[v][i] = adjacencyMatrix[i][v] = 1;
20.                 Integer val = degreeSequence.get(i);
21.                 if (val > 0)
22.                     degreeSequence.set(i, val - 1);
23.                 e--;
24.             }
25.         }
26.     }
27.
28.     public void generateGraph()
29.     {
30.         adjacencyMatrix = new Integer[degreeSequence.size()][degreeSequence
31.             .size()];
32.         for (int i = 0; i < adjacencyMatrix.length; i++)
33.         {
34.             for (int j = 0; j < adjacencyMatrix.length; j++)
35.             {
36.                 adjacencyMatrix[i][j] = 0;
37.             }
38.         }
39.         for (int i = 0; i < degreeSequence.size(); i++)
40.         {
41.             Integer e = degreeSequence.get(i);
42.             degreeSequence.set(i, 0);
43.             addEdges(i, e);
44.         }
45.     }
46. }
```

```

45. }
46.
47. public void printGraph()
48. {
49.     System.out.println("The matrix form of graph: ");
50.     for (int i = 0; i < adjacencyMatrix.length; i++)
51.     {
52.         for (int j = 0; j < adjacencyMatrix.length; j++)
53.         {
54.             System.out.print(adjacencyMatrix[i][j] + " ");
55.         }
56.         System.out.println();
57.     }
58. }
59.
60. public static void main(String[] args)
61. {
62.     Scanner sc = new Scanner(System.in);
63.     System.out.println("Enter the number of vertices: ");
64.     Integer n = sc.nextInt();
65.     System.out
66.         .println("Enter the Degree Sequence: <Degree sequence is always in non-increasing order>");
67.     GraphUsingDegreeSequence gds = new GraphUsingDegreeSequence();
68.     gds.degreeSequence = new ArrayList<Integer>();
69.     while (n > 0)
70.     {
71.         gds.degreeSequence.add(sc.nextInt());
72.         n--;
73.     }
74.     System.out.println("Entered degree sequence: "
75.         + gds.degreeSequence.toString());
76.     gds.generateGraph();
77.     gds.printGraph();
78.     sc.close();
79. }
80. }
```

Output:

advertisement

```
$ javac GraphUsingDegreeSequence.java
$ java GraphUsingDegreeSequence
```

Enter the number of vertices:

7

Enter the Degree Sequence: <Degree sequence is always in non-increasing order>

5 3 3 2 2 1 0

Entered degree sequence: [5, 3, 3, 2, 2, 1, 0]

The matrix form of graph:

```
0 1 1 1 1 1 0
1 0 1 1 0 0 0
1 1 0 0 1 0 0
1 1 0 0 0 0 0
1 0 1 0 0 0 0
1 0 0 0 0 0 0
0 0 0 0 0 0 0
```

301. Java Program to Implement Heap's Algorithm for Permutation of N Numbers

[« Prev](#)

[Next »](#)

This is a java program to find permutation of N numbers using Heap's Algorithm. Heap's algorithm is an algorithm used for generating all possible permutations of some given length.

Here is the source code of the Java Program to Implement Heap's Algorithm for Permutation of N Numbers. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.combinatorial;
3.
4. import java.util.Arrays;
5. import java.util.Scanner;
6.
7. public class HeapsPermutation
8. {
9.     private static void swap(int[] v, int i, int j)
10.    {
11.        int t = v[i];
12.        v[i] = v[j];
13.        v[j] = t;
14.    }
15.
16.    public void permute(int[] v, int n)
17.    {
18.        if (n == 1)
19.        {
20.            System.out.println(Arrays.toString(v));
21.        }
22.        else
23.        {
24.            for (int i = 0; i < n; i++)
25.            {
26.                permute(v, n - 1);
27.                if (n % 2 == 1)
28.                {
29.                    swap(v, 0, n - 1);
30.                }
31.                else
32.                {
33.                    swap(v, i, n - 1);
34.                }
35.            }
36.        }
37.    }
38.
39.    public static void main(String[] args)
40.    {
41.        System.out.println("Enter the number of elements in a sequence: ");
42.        Scanner sc = new Scanner(System.in);
43.        int n = sc.nextInt();
44.        System.out.println("Enter the sequence: ");
45.        int sequence[] = new int[n];
46.        for (int i = 0; i < n; i++)
47.        {
48.            sequence[i] = sc.nextInt();
49.        }
50.        new HeapsPermutation().permute(sequence, n);
51.        sc.close();
52.    }
53. }
```

```
52. }  
53. }
```

Output:

```
$ javac HeapsPermutation.java  
$ java HeapsPermutation
```

Enter the number of elements in a sequence:

5

Enter the sequence:

```
2 4 7 3 8  
[2, 4, 7, 3, 8]  
[4, 2, 7, 3, 8]  
[7, 2, 4, 3, 8]  
[2, 7, 4, 3, 8]  
[4, 7, 2, 3, 8]  
[7, 4, 2, 3, 8]  
[3, 4, 7, 2, 8]  
[4, 3, 7, 2, 8]  
[7, 3, 4, 2, 8]  
[3, 7, 4, 2, 8]  
[4, 7, 3, 2, 8]  
[7, 4, 3, 2, 8]  
[3, 2, 7, 4, 8]  
[2, 3, 7, 4, 8]  
[7, 3, 2, 4, 8]  
[3, 7, 2, 4, 8]  
[2, 7, 3, 4, 8]  
[7, 2, 3, 4, 8]  
[3, 2, 4, 7, 8]  
[2, 3, 4, 7, 8]  
[4, 3, 2, 7, 8]  
[3, 4, 2, 7, 8]  
[2, 4, 3, 7, 8]  
[4, 2, 3, 7, 8]  
[8, 2, 4, 7, 3]  
[2, 8, 4, 7, 3]  
[4, 8, 2, 7, 3]  
[8, 4, 2, 7, 3]  
[2, 4, 8, 7, 3]  
[4, 2, 8, 7, 3]  
[7, 2, 4, 8, 3]  
[2, 7, 4, 8, 3]  
[4, 7, 2, 8, 3]  
[7, 4, 2, 8, 3]  
[2, 4, 7, 8, 3]  
[4, 2, 7, 8, 3]  
[7, 8, 4, 2, 3]  
[8, 7, 4, 2, 3]  
[4, 7, 8, 2, 3]  
[7, 4, 8, 2, 3]  
[8, 4, 7, 2, 3]  
[4, 8, 7, 2, 3]  
[7, 8, 2, 4, 3]  
[8, 7, 2, 4, 3]  
[2, 7, 8, 4, 3]  
[7, 2, 8, 4, 3]  
[8, 2, 7, 4, 3]  
[2, 8, 7, 4, 3]  
[3, 8, 2, 4, 7]  
[8, 3, 2, 4, 7]  
[2, 3, 8, 4, 7]  
[3, 2, 8, 4, 7]  
[8, 2, 3, 4, 7]  
[2, 8, 3, 4, 7]  
[4, 8, 2, 3, 7]
```

[8, 4, 2, 3, 7]
[2, 4, 8, 3, 7]
[4, 2, 8, 3, 7]
[8, 2, 4, 3, 7]
[2, 8, 4, 3, 7]
[4, 3, 2, 8, 7]
[3, 4, 2, 8, 7]
[2, 4, 3, 8, 7]
[4, 2, 3, 8, 7]
[3, 2, 4, 8, 7]
[2, 3, 4, 8, 7]
[4, 3, 8, 2, 7]
[3, 4, 8, 2, 7]
[8, 4, 3, 2, 7]
[4, 8, 3, 2, 7]
[3, 8, 4, 2, 7]
[8, 3, 4, 2, 7]
[7, 3, 8, 2, 4]
[3, 7, 8, 2, 4]
[8, 7, 3, 2, 4]
[7, 8, 3, 2, 4]
[3, 8, 7, 2, 4]
[8, 3, 7, 2, 4]
[2, 3, 8, 7, 4]
[3, 2, 8, 7, 4]
[8, 2, 3, 7, 4]
[2, 8, 3, 7, 4]
[3, 8, 2, 7, 4]
[8, 3, 2, 7, 4]
[2, 7, 8, 3, 4]
[7, 2, 8, 3, 4]
[8, 2, 7, 3, 4]
[2, 8, 7, 3, 4]
[7, 8, 2, 3, 4]
[8, 7, 2, 3, 4]
[2, 7, 3, 8, 4]
[7, 2, 3, 8, 4]
[3, 2, 7, 8, 4]
[2, 3, 7, 8, 4]
[7, 3, 2, 8, 4]
[3, 7, 2, 8, 4]
[4, 7, 3, 8, 2]
[7, 4, 3, 8, 2]
[3, 4, 7, 8, 2]
[4, 3, 7, 8, 2]
[7, 3, 4, 8, 2]
[3, 7, 4, 8, 2]
[8, 7, 3, 4, 2]
[7, 8, 3, 4, 2]
[3, 8, 7, 4, 2]
[8, 3, 7, 4, 2]
[7, 3, 8, 4, 2]
[3, 7, 8, 4, 2]
[8, 4, 3, 7, 2]
[4, 8, 3, 7, 2]
[3, 8, 4, 7, 2]
[8, 3, 4, 7, 2]
[4, 3, 8, 7, 2]
[3, 4, 8, 7, 2]
[8, 4, 7, 3, 2]
[4, 8, 7, 3, 2]
[7, 8, 4, 3, 2]
[8, 7, 4, 3, 2]
[4, 7, 8, 3, 2]
[7, 4, 8, 3, 2]

302. Java Program to Find ith Largest Number from a Given List Using Order-Statistic Algorithm

[« Prev](#)

[Next »](#)

This is a java program to find the ith largest element from a list using order-statistic algorithm. This version of the code uses quick sort partitioning technique.

Here is the source code of the Java Program to Find ith Largest Number from a Given List Using Order-Statistic Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.combinatorial;
3.
4. import java.util.Scanner;
5.
6. public class KthLargestOrderStatistics
7. {
8.     public static int partition(int[] array, int first, int last)
9.     {
10.         int pivot = array[first];
11.         int pivotPosition = first++;
12.         while (first <= last)
13.         {
14.             // scan for values less than the pivot
15.             while ((first <= last) && (array[first] < pivot))
16.             {
17.                 first++;
18.             }
19.             // scan for values greater than the pivot
20.             while ((last >= first) && (array[last] >= pivot))
21.             {
22.                 last--;
23.             }
24.             if (first > last)
25.             {
26.                 // swap the last uncoformed
27.                 // element with the pivot
28.                 swap(array, pivotPosition, last);
29.             }
30.             else
31.             {
32.                 // swap uncoformed elements:
33.                 // first that was not lesser than the pivot
34.                 // and last that was not larger than the pivot
35.                 swap(array, first, last);
36.             }
37.         }
38.         return last;
39.     }
40.
41.     private static void swap(int[] array, int first, int last)
42.     {
43.         int temp;
44.         temp = array[first];
45.         array[first] = array[last];
46.         array[last] = temp;
47.     }
48.
49.     public static int orderStatistic(int[] array, int k, int first, int last)
50.     {
```

```

51.     int pivotPosition = partition(array, first, last);
52.     if (pivotPosition == k - 1)
53.     {
54.         return array[k - 1];
55.     }
56.     if (k - 1 < pivotPosition)
57.     {
58.         return orderStatistics(array, k, first, pivotPosition - 1);
59.     }
60.     else
61.     {
62.         return orderStatistics(array, k, pivotPosition + 1, last);
63.     }
64. }
65.
66. // iterative version
67. private static int orderStatistics(int[] array, int k, int first, int last)
68. {
69.     int pivotPosition = partition(array, first, last);
70.     while (pivotPosition != k - 1)
71.     {
72.         if (k - 1 < pivotPosition)
73.         {
74.             last = pivotPosition - 1;
75.         }
76.         else
77.         {
78.             first = pivotPosition + 1;
79.         }
80.         pivotPosition = partition(array, first, last);
81.     }
82.     return array[k - 1];
83. }
84.
85. public static int kthSmallest(int[] array, int k)
86. {
87.     return orderStatistic(array, k, 0, array.length - 1);
88. }
89.
90. public static int kthLargest(int[] array, int k)
91. {
92.     return orderStatistic(array, array.length - k + 1, 0, array.length - 1);
93. }
94.
95. public static void main(String[] args)
96. {
97.     Scanner sc = new Scanner(System.in);
98.     System.out.println("Enter the number of elements in the sequence: ");
99.     int n = sc.nextInt();
100.    int[] sequence = new int[n];
101.    System.out.println("Enter the elements of the sequence: ");
102.    for (int i = 0; i < sequence.length; i++)
103.    {
104.        sequence[i] = sc.nextInt();
105.    }
106.    System.out
107.        .println("Enter the kth index to be returned as kth largest element of the sequence:");
108.    int k = sc.nextInt();
109.    System.out.println("Kth largest:" + kthLargest(sequence, k));
110.    sc.close();
111. }
112. }
```

Output:

advertisement

```
$ javac KthLargestOrderStatistics.java  
$ java KthLargestOrderStatistics
```

Enter the number of elements in the sequence:

10

Enter the elements of the sequence:

2 5 6 7 4 7 9 5 8 1

Enter the kth index to be returned as kth largest element of the sequence:

4

Kth largest:7

303. Java Program to Perform Searching Based on Locality of Reference

[« Prev](#)

[Next »](#)

This is a java program to perform searching based on locality of reference. The Locality of Reference principle says that if an element of a list is accessed it might also be accessed in near future. So we store it to the front of the list.

Here is the source code of the Java Program to Perform Searching Based on Locality of Reference. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. /*
3.  * Follows recently accessed elements should be referred more so kept on top of
4.  * the list
5. */
6. package com.sanfoundry.combinatorial;
7.
8. import java.util.LinkedList;
9. import java.util.List;
10.import java.util.Random;
11.import java.util.Scanner;
12.
13.public class LocalityBasedSearching
14.
15. public static void main(String[] args)
16. {
17.     List<Integer> items = new LinkedList<Integer>();
18.     Integer n = 10;
19.     Scanner sc = new Scanner(System.in);
20.     Random rand = new Random();
21.     while (n > 0)
22.     {
23.         items.add(rand.nextInt(100));
24.         n--;
25.     }
26.     System.out.println(items.toString());
27.     boolean flag = true;
28.     boolean found = false;
29.     Integer numberofInstance;
30.     while (flag == true)
31.     {
32.         numberofInstance = 0;
33.         System.out.println("Enter the element to find: ");
34.         Integer search = sc.nextInt();
35.         for (int i = 0; i < items.size(); i++)
36.         {
37.             if (items.get(i).equals(search))
38.             {
39.                 found = true;
40.                 System.out.println("Element found at index " + i
41.                     + "\nReordering list..");
42.                 // Integer temp = items.get(numberofInstance);
43.                 // items.set(numberofInstance, search);
44.                 items.add(numberofInstance, search);
45.                 items.remove(i + 1);
46.                 // items.set(i, temp);
47.                 System.out.println("Reordered list: " + items.toString());
48.                 numberofInstance++;
49.                 // break;
50.             }
51.         }
```

```
52.     if (found == false)
53.     {
54.         System.out.println("No such element found.");
55.     }
56.     System.out.println("Do you want to continue? <true>/<false>");
57.     flag = sc.nextBoolean();
58.   }
59.   sc.close();
60. }
61.}
```

Output:

advertisement

```
$ javac LocalityBasedSearching.java
$ java LocalityBasedSearching
```

```
[52, 94, 58, 8, 78, 0, 30, 81, 16, 58]
Enter the element to find:
8
Element found at index 3
Reordering list...
Reordered list: [8, 52, 94, 58, 78, 0, 30, 81, 16, 58]
Do you want to continue? <true>/<false>
true
Enter the element to find:
58
Element found at index 3
Reordering list...
Reordered list: [58, 8, 52, 94, 78, 0, 30, 81, 16, 58]
Element found at index 9
Reordering list...
Reordered list: [58, 58, 8, 52, 94, 78, 0, 30, 81, 16]
Do you want to continue? <true>/<false>
true
Enter the element to find:
94
Element found at index 4
Reordering list...
Reordered list: [94, 58, 58, 8, 52, 78, 0, 30, 81, 16]
Do you want to continue? <true>/<false>
false
```

304. Java Program to Create the Prüfer Code for a Tree

[« Prev](#)

[Next »](#)

This is a java program to create Prüfer code for a tree. In combinatorial mathematics, the Prüfer sequence (also Prüfer code or Prüfer numbers) of a labeled tree is a unique sequence associated with the tree. The sequence for a tree on n vertices has length $n - 2$, and can be generated by a simple iterative algorithm.

Here is the source code of the Java Program to Create the Prüfer Code for a Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.combinatorial;
3.
4. import java.util.ArrayList;
5. import java.util.Arrays;
6. import java.util.Collections;
7. import java.util.LinkedHashSet;
8. import java.util.List;
9. import java.util.Random;
10.import java.util.Scanner;
11.import java.util.Set;
12.
13.public class PrüferCode
14.{
15.    public static List<Integer>[] getRandomTree2(int n, Random rnd)
16.    {
17.        @SuppressWarnings("unchecked")
18.        List<Integer>[] t = new List[n];
19.        for (int i = 0; i < n; i++)
20.            t[i] = new ArrayList<>();
21.        int[] p = new int[n];
22.        for (int i = 0, j; i < n; j = rnd.nextInt(i + 1), p[i] = p[j], p[j] = i, i++)
23.            ; // random permutation
24.        for (int i = 1; i < n; i++)
25.        {
26.            int parent = p[rnd.nextInt(i)];
27.            t[parent].add(p[i]);
28.            t[p[i]].add(parent);
29.        }
30.        return t;
31.    }
32.
33.    public static List<Integer>[] prüferCode2Tree(int[] prüferCode)
34.    {
35.        int n = prüferCode.length + 2;
36.        @SuppressWarnings("unchecked")
37.        List<Integer>[] tree = new List[n];
38.        for (int i = 0; i < n; i++)
39.            tree[i] = new ArrayList<>();
40.        int[] degree = new int[n];
41.        Arrays.fill(degree, 1);
42.        for (int v : prüferCode)
43.            ++degree[v];
44.        int ptr = 0;
45.        while (degree[ptr] != 1)
46.            ++ptr;
47.        int leaf = ptr;
48.        for (int v : prüferCode)
49.        {
50.            tree[leaf].add(v);
```

```

51.     tree[v].add(leaf);
52.     --degree[leaf];
53.     --degree[v];
54.     if (degree[v] == 1 && v < ptr)
55.     {
56.         leaf = v;
57.     }
58.     else
59.     {
60.         for (++ptr; ptr < n && degree[ptr] != 1; ++ptr)
61.             ;
62.         leaf = ptr;
63.     }
64. }
65. for (int v = 0; v < n - 1; v++)
66. {
67.     if (degree[v] == 1)
68.     {
69.         tree[v].add(n - 1);
70.         tree[n - 1].add(v);
71.     }
72. }
73. return tree;
74. }
75.
76. public static int[] tree2PruferCode(List<Integer>[] tree)
77. {
78.     int n = tree.length;
79.     int[] parent = new int[n];
80.     parent[n - 1] = -1;
81.     pruferDfs(tree, parent, n - 1);
82.     int[] degree = new int[n];
83.     int ptr = -1;
84.     for (int i = 0; i < n; ++i)
85.     {
86.         degree[i] = tree[i].size();
87.         if (degree[i] == 1 && ptr == -1)
88.             ptr = i;
89.     }
90.     int[] res = new int[n - 2];
91.     int leaf = ptr;
92.     for (int i = 0; i < n - 2; ++i)
93.     {
94.         int next = parent[leaf];
95.         res[i] = next;
96.         --degree[next];
97.         if (degree[next] == 1 && next < ptr)
98.         {
99.             leaf = next;
100.        }
101.        else
102.        {
103.            ++ptr;
104.            while (ptr < n && degree[ptr] != 1)
105.                ++ptr;
106.            leaf = ptr;
107.        }
108.    }
109.    return res;
110. }
111.
112. static void pruferDfs(List<Integer>[] tree, int[] parent, int v)
113. {
114.     for (int i = 0; i < tree[v].size(); ++i)
115.     {
116.         int to = tree[v].get(i);

```

```

117.     if (to != parent[v])
118.     {
119.         parent[to] = v;
120.         pruferDfs(tree, parent, to);
121.     }
122. }
123. }
124.
125. // precondition: n >= 2
126. public static List<Integer>[] getRandomTree(int V, Random rnd)
127. {
128.     int[] a = new int[V - 2];
129.     for (int i = 0; i < a.length; i++)
130.     {
131.         a[i] = rnd.nextInt(V);
132.     }
133.     return pruferCode2Tree(a);
134. }
135.
136. // precondition: V >= 2, V-1 <= E <= V*(V-1)/2
137. public static List<Integer>[] getRandomUndirectedConnectedGraph(int V,
138.                     int E, Random rnd)
139. {
140.     List<Integer>[] g = getRandomTree(V, rnd);
141.     Set<Long> edgeSet = new LinkedHashSet<>();
142.     for (int i = 0; i < V; i++)
143.     {
144.         for (int j = i + 1; j < V; j++)
145.         {
146.             edgeSet.add(((long) i << 32) + j);
147.         }
148.     }
149.     for (int i = 0; i < V; i++)
150.     {
151.         for (int j : g[i])
152.         {
153.             edgeSet.remove(((long) i << 32) + j);
154.         }
155.     }
156.     List<Long> edges = new ArrayList<>(edgeSet);
157.     for (int x : getRandomArrangement(edges.size(), E - (V - 1), rnd))
158.     {
159.         long e = edges.get(x);
160.         int u = (int) (e >>> 32);
161.         int v = (int) e;
162.         g[u].add(v);
163.         g[v].add(u);
164.     }
165.     for (int i = 0; i < V; i++)
166.         Collections.sort(g[i]);
167.     return g;
168. }
169.
170. // precondition: V >= 2, V-1 <= E <= V*(V-1)/2
171. public static List<Integer>[] getRandomUndirectedConnectedGraph2(int V,
172.                     int E, Random rnd)
173. {
174.     List<Integer>[] g = getRandomTree(V, rnd);
175.     Set<Long> edgeSet = new LinkedHashSet<>();
176.     for (int i = 0; i < V; i++)
177.     {
178.         for (int j : g[i])
179.         {
180.             edgeSet.add(((long) i << 32) + j);
181.         }
182.     }

```

```

183.     for (int i = 0; i < E - (V - 1); i++)
184.     {
185.         int u;
186.         int v;
187.         long edge;
188.         while (true)
189.         {
190.             u = rnd.nextInt(V);
191.             v = rnd.nextInt(V);
192.             edge = ((long) u << 32) + v;
193.             if (u < v && !edgeSet.contains(edge))
194.                 break;
195.             }
196.             edgeSet.add(edge);
197.             g[u].add(v);
198.             g[v].add(u);
199.         }
200.         for (int i = 0; i < V; i++)
201.             Collections.sort(g[i]);
202.         return g;
203.     }
204.
205. static int[] getRandomArrangement(int n, int m, Random rnd)
206. {
207.     int[] res = new int[n];
208.     for (int i = 0; i < n; i++)
209.     {
210.         res[i] = i;
211.     }
212.     for (int i = 0; i < m; i++)
213.     {
214.         int j = n - 1 - rnd.nextInt(n - i);
215.         int t = res[i];
216.         res[i] = res[j];
217.         res[j] = t;
218.     }
219.     return Arrays.copyOf(res, m);
220. }
221.
222. static void checkGraph(int V, int E, Random rnd)
223. {
224.     List<Integer>[] g = getRandomUndirectedConnectedGraph(V, E, rnd);
225.     int n = g.length;
226.     int[][] a = new int[n][n];
227.     int edges = 0;
228.     for (int i = 0; i < n; i++)
229.     {
230.         for (int j : g[i])
231.         {
232.             ++a[i][j];
233.             ++edges;
234.         }
235.     }
236.     if (edges != 2 * E)
237.     {
238.         throw new RuntimeException();
239.     }
240.     for (int i = 0; i < n; i++)
241.     {
242.         if (a[i][i] != 0)
243.         {
244.             throw new RuntimeException();
245.         }
246.         for (int j = 0; j < n; j++)
247.         {
248.             if (a[i][j] != a[j][i] || a[i][j] != 0 && a[i][j] != 1)

```

```

249.         {
250.             throw new RuntimeException();
251.         }
252.     }
253. }
255.
256. public static void main(String[] args)
257. {
258.     Scanner sc = new Scanner(System.in);
259.     System.out.println("Enter the length of code: ");
260.     int n = sc.nextInt();
261.     int[] code = new int[n];
262.     for (int i = 0; i < n; i++)
263.     {
264.         code[i] = sc.nextInt();
265.     }
266.     List<Integer>[] tree = pruferCode2Tree(code);
267.     Random rnd = new Random(1);
268.     for (int step = 0; step < 1000; step++)
269.     {
270.         int V = rnd.nextInt(50) + 2;
271.         checkGraph(V, V - 1, rnd);
272.         checkGraph(V, V * (V - 1) / 2, rnd);
273.         checkGraph(V, rnd.nextInt(V * (V - 1) / 2 - (V - 1) + 1) + V - 1,
274.                     rnd);
275.     }
276.     System.out.println("Prufer code to tree conversion: "
277.                         + Arrays.toString(tree));
278.     System.out.println("Tree to prufer code conversion: "
279.                         + Arrays.toString(tree2PruferCode(tree)));
280.     sc.close();
281. }
282.

```

Output:

advertisement

```
$ javac PruferCode.java
$ java PruferCode
```

```
Enter the length of code:
4
2 3 4 3
Prufer code to tree conversion: [[2], [3], [0, 4], [1, 4, 5], [2, 3], [3]]
Tree to prufer code conversion: [2, 3, 4, 3]
```

```
Enter the length of code:
3
2 4 3
Prufer code to tree conversion: [[2], [4], [0, 3], [2, 4], [1, 3]]
Tree to prufer code conversion: [2, 4, 3]
```

```
Enter the length of code:
5
3 4 5 1 6
Prufer code to tree conversion: [[3], [4, 6], [4], [0, 5], [2, 1], [3, 6], [1, 5]]
Tree to prufer code conversion: [3, 4, 5, 1, 6]
```

305. Java Program to Implement Quick Sort with Given Complexity Constraint

[« Prev](#)

[Next »](#)

This is a java program to perform quick sort with complexity constraint of time less than n^2 .

Here is the source code of the Java Program to Implement Quick Sort with Given Complexity Constraint. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.combinatorial;
3. 
4. import java.util.Random;
5. 
6. public class QuickSortComplexityConstraint
7. {
8.     public static int N = 20;
9.     public static int[] sequence = new int[N];
10. 
11.    public static void QuickSort(int left, int right)
12.    {
13.        if (right - left <= 0)
14.            return;
15.        else
16.        {
17.            Random rand = new Random();
18.            int pivotIndex = left + rand.nextInt(right - left + 1);
19.            swap(pivotIndex, right);
20.            int pivot = sequence[right];
21.            int partition = partitionIt(left, right, pivot);
22.            QuickSort(left, partition - 1);
23.            QuickSort(partition + 1, right);
24.        }
25.    }
26. 
27.    public static int partitionIt(int left, int right, long pivot)
28.    {
29.        int leftPtr = left - 1;
30.        int rightPtr = right;
31.        while (true)
32.        {
33.            while (sequence[++leftPtr] < pivot)
34.                ;
35.            while (rightPtr > 0 && sequence[--rightPtr] > pivot)
36.                ;
37.            if (leftPtr >= rightPtr)
38.                break;
39.            else
40.                swap(leftPtr, rightPtr);
41.        }
42.        swap(leftPtr, right);
43.        return leftPtr;
44.    }
45. 
46.    public static void swap(int dex1, int dex2)
47.    {
48.        int temp = sequence[dex1];
49.        sequence[dex1] = sequence[dex2];
50.        sequence[dex2] = temp;
51.    }
52. 
53.    static void printSequence(int[] sorted_sequence)
```

```
54.  {
55.    for (int i = 0; i < sorted_sequence.length; i++)
56.      System.out.print(sorted_sequence[i] + " ");
57.  }
58.
59. public static void main(String args[])
60. {
61.   System.out
62.     .println("Sorting of randomly generated numbers using QUICK SORT with complexity less than n^2");
63.   Random random = new Random();
64.   for (int i = 0; i < N; i++)
65.     sequence[i] = Math.abs(random.nextInt(100));
66.   System.out.println("\nOriginal Sequence: ");
67.   printSequence(sequence);
68.   System.out.println("\nSorted Sequence: ");
69.   QuickSort(0, N - 1);
70.   printSequence(sequence);
71. }
72.
```

Output:

advertisement

```
$ javac QuickSortComplexityConstraint.java
$ java QuickSortComplexityConstraint
```

Sorting of randomly generated numbers using QUICK SORT with complexity less than n^2

Original Sequence:

29 19 67 48 23 99 72 40 23 93 0 79 70 87 43 24 56 67 51 71

Sorted Sequence:

0 19 23 23 24 29 40 43 48 51 56 67 67 70 71 72 79 87 93 99

306. Java Program to Generate a Sequence of N Characters for a Given Specific Case

[« Prev](#)

[Next »](#)

This is a java program to generate sequence of N characters randomly.

Here is the source code of the Java Program to Generate a Sequence of N Characters for a Given Specific Case. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. // Specific case is that characters are generated randomly
3. package com.sanfoundry.combinatorial;
4.
5. import java.util.Scanner;
6.
7. public class SequenceOfNCharacters
8. {
9.     public static Integer randomInt(Integer low, Integer high)
10.    {
11.        return (int) (Math.floor(Math.random() * (high - low + 1)) + low);
12.    }
13.
14.    public static Character randomChar(String str)
15.    {
16.        return str.charAt(randomInt(0, str.length() - 1));
17.    }
18.
19.    public static String generateRandSeq(Integer length, String src)
20.    {
21.        String seq = "";
22.        for (int i = 1; i <= length; i = i + 1)
23.        {
24.            seq += randomChar(src);
25.        }
26.        return seq;
27.    }
28.
29.    public static void main(String[] args)
30.    {
31.        String src = "abcdefghijklmnopqrstuvwxyz";
32.        Scanner sc = new Scanner(System.in);
33.        System.out.println("Enter the number of sequences to be generated: ");
34.        int numberOfSequence = sc.nextInt();
35.        System.out.println("Enter the length of each sequence: ");
36.        int length = sc.nextInt();
37.        for (int i = 0; i < numberOfSequence; i++)
38.        {
39.            System.out.println(generateRandSeq(length, src));
40.        }
41.        sc.close();
42.    }
43.}
```

Output:

advertisement

```
$ javac SequenceOfNCharacters.java
$ java SequenceOfNCharacters
```

Enter the number of sequences to be generated:

4

Enter the length of each sequence:

5

qgpnt
kdxyr
ynhmf
wambi

Enter the number of sequences to be generated:

3

Enter the length of each sequence:

8

ilhddizq
evmpejxv
malvlhja

307. Create Java Applet to Simulate Any Sorting Technique

[« Prev](#)

[Next »](#)

This is a java program to create an applet to simulate a sorting technique. This applet demonstrates Bubble Sort.

Here is the source code of the Create Java Applet to Simulate Any Sorting Technique. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. package com.sanfoundry.combinatorial;
2.
3. import java.applet.Applet;
4. import java.awt.BorderLayout;
5. import java.awt.Button;
6. import java.awt.Canvas;
7. import java.awt.Choice;
8. import java.awt.Color;
9. import java.awt.Dimension;
10.import java.awt.Event;
11.import java.awt.FlowLayout;
12.import java.awt.Font;
13.import java.awt.FontMetrics;
14.import java.awt.Graphics;
15.import java.awt.Image;
16.import java.awt.Panel;
17.import java.awt.Point;
18.
19.@SuppressWarnings("deprecation")
20.class ExplainBox extends Canvas
21.{
22.    private static final long serialVersionUID = 1L;
23.    static final int marginX = 6;
24.    static final int marginY = 3;
25.    String text = "";
26.
27.    public ExplainBox()
28.    {
29.        setFont(new Font("TimesRoman", Font.PLAIN, 12));
30.    }
31.
32.    public void setText(String text)
33.    {
34.        this.text = text;
35.        invalidate();
36.    }
37.
38.    public void validate()
39.    {
40.        FontMetrics metrics = getFontMetrics(getFont());
41.        int baseLine = metrics.getAscent();
42.        int lineHeight = baseLine + metrics.getDescent();
43.        int width = metrics.stringWidth(text);
44.        Point corner = location();
45.        reshape(corner.x, corner.y, width + 2 * marginX, lineHeight + 2
46.                * marginY);
47.    }
48.
49.    public void paint(Graphics g)
50.    {
51.        g.setColor(Color.black);
52.        Dimension size = size();
53.        g.drawRect(0, 0, size.width - 1, size.height - 1);
```

```

54.     FontMetrics metrics = getFontMetrics(getFont());
55.     g.drawString(text, marginX, marginY + metrics.getAscent());
56. }
57.
58. public boolean mouseExit(Event event, int x, int y)
59. {
60.     return true;
61. }
62.}
63.
64.@SuppressWarnings("deprecation")
65.class CodePanel extends Panel
66.
67. private static final long serialVersionUID = 1L;
68. static final int marginX = 15;
69. static final int marginY = 20;
70. static final int offsetX = 1;
71. static final int offsetY = 1;
72. static final int none = -1;
73. String code[]; // Array to hold the source code
74. String explanations[]; // Array to hold the explanation of source code
75. Font font = new Font("TimesRoman", Font.PLAIN, 16);
76. int lineHeight;
77. int baseLine;
78. int maxWidth = 0;
79. int highlightedLine = none;
80. ExplainBox explaination = new ExplainBox();
81. Applet applet;
82.
83. public CodePanel(String code[], String explanations[], Applet applet)
84. {
85.     this.code = code;
86.     this.explanations = explanations;
87.     this.applet = applet;
88.     setBackground(Color.white); // Set the background of code panel to be
89.                         // white
90.     explaination.setBackground(Color.lightGray);
91.     explaination.setForeground(Color.lightGray);
92.     add(explaination);
93.     explaination.hide(); // Hide explaination until the code line is clicked
94. }
95.
96. public Dimension preferredSize()
97. {
98.     return new Dimension(280, 300);
99. }
100.
101. public void addNotify()
102. {
103.     super.addNotify();
104.     setFont(font);
105.     FontMetrics metrics = getFontMetrics(font);
106.     baseLine = metrics.getAscent();
107.     lineHeight = baseLine + metrics.getDescent();
108.     for (int i = 0; i < code.length; i++)
109.     {
110.         maxWidth = Math.max(maxWidth, metrics.stringWidth(code[i]));
111.     }
112. }
113.
114. public void paint(Graphics g)
115. {
116.     int y = marginY + baseLine;
117.     for (int i = 0; i < code.length; i++, y += lineHeight)
118.     {
119.         setBackground(Color.white);

```

```

120.         g.drawString(code[i], marginX, y);
121.     }
122.     highlightLine(highlightedLine);
123. }
124.
125. public void reset()
126. {
127.     if (highlightedLine != none)
128.     {
129.         colorLine(highlightedLine, Color.white);
130.     }
131.     highlightedLine = none;
132. }
133.
134. public void highlightLine(int line)
135. {
136.     if (highlightedLine != none)
137.     {
138.         colorLine(highlightedLine, Color.white);
139.     }
140.     highlightedLine = line;
141.     if (highlightedLine != none)
142.     {
143.         colorLine(highlightedLine, Color.pink);
144.     }
145. }
146.
147. public void colorLine(int line, Color color)
148. {
149.     Graphics g = getGraphics();
150.     int y = marginY + line * lineHeight;
151.     g.setColor(color);
152.     g.fillRect(0, y, size().width - 1, lineHeight);
153.     g.setColor(Color.black);
154.     g.drawString(code[line], marginX, y + baseLine);
155. }
156.
157. public boolean mouseExit(Event event, int x, int y)
158. {
159.     explanation.hide();
160.     validate();
161.     return true;
162. }
163.
164. public boolean mouseUp(Event event, int x, int y)
165. {
166.     int line = (y - marginY) / lineHeight;
167.     if ((line <= explanations.length) || (explanations[line].equals("")))
168.     {
169.         explanation.setText(explanations[line]);
170.         explanation.setBackground(Color.lightGray);
171.         explanation.validate();
172.         explanation.show();
173.     }
174.     else
175.     {
176.         explanation.hide();
177.     }
178.     validate();
179.     explanation.move(marginX + offsetX, marginY + offsetY + (line + 1)
180.                     * lineHeight);
181.     return true;
182. }
183. }
184.
185.@SuppressWarnings("deprecation")

```

```

186.class Algorithm extends Thread
187.{
188.    CodePanel codeDisplay; // Code Panel
189.    static int granularity; // Granularity Level
190.    SortingApplet applet; // Bubble Sort Applet
191.    Animation animation; // Animation Canvas
192.    public static int indexi = 0; // Loop Index
193.    public static int indexj = 0; // Loop Index
194.    public static int flag = -1;
195.
196.    public Algorithm(CodePanel codeDisplay, int granularity, SortingApplet applet,
197.                    Animation animation)
198.    {
199.        this.codeDisplay = codeDisplay;
200.        this.applet = applet;
201.        Algorithm.granularity = granularity;
202.        this.animation = animation;
203.    }
204.
205.    void setGranularity(int granularity)
206.    {
207.        Algorithm.granularity = granularity;
208.    }
209.
210.    public void run()
211.    {
212.        int line = 0; // Line Number
213.        visualize(line, 2); // Visualize current line
214.        indexi = SortingApplet.SourceData.length - 1; // Set loop index value
215.        Algorithm.flag = 1; // Set execution status
216.        animation.repaint(); // Refresh animation canvas
217.        int forLoopLine1 = line; // Mark the line # of first for loop
218.        while (true)
219.        {
220.            if (!(indexi >= 1))
221.                break;
222.            visualize(++line, 2);
223.            indexj = 0;
224.            animation.repaint();
225.            int forLoopLine2 = line; // Mark the line # of second for loop
226.            while (true)
227.            {
228.                if (!(indexj <= (indexi - 1)))
229.                    break;
230.                visualize(++line, 2);
231.                animation.repaint();
232.                if (SortingApplet.SourceData[indexj] > SortingApplet.SourceData[indexj + 1])
233.                {
234.                    // switch the two array elements
235.                    visualize(++line, 2);
236.                    int temp = SortingApplet.SourceData[indexj];
237.                    animation.repaint();
238.                    visualize(++line, 2);
239.                    SortingApplet.SourceData[indexj] = SortingApplet.SourceData[indexj + 1];
240.                    animation.repaint();
241.                    visualize(++line, 2);
242.                    SortingApplet.SourceData[indexj + 1] = temp;
243.                    animation.repaint();
244.                }
245.                line = forLoopLine2; // Set line # to be the second for loop
246.                visualize(line, 1);
247.                animation.repaint();
248.                indexj++;
249.            }
250.            line = forLoopLine1; // Set line # to be the first for loop
251.            visualize(line, 1);

```

```

252.     indexi--;
253.     animation.repaint();
254. }
255. Algorithm.flag = -1; // After execution finished, set flag back to -1
256. animation.repaint();
257. try
258. {
259.     sleep(1000);
260. }
261. catch (Exception e)
262. {
263. }
264. applet.sortButton.setLabel(" Sort ");
265. applet.sortButton.disable();
266. applet.stopButton.disable();
267. applet.resetButton.enable();
268. visualize(0, 0); // After execution finished, highlight the first line
269. applet.finished();
270. }
271.
272. void visualize(int line, int level)
273. {
274.     codeDisplay.highlightLine(line); // Highlight the current line
275.     codeDisplay.repaint();
276.     if (level > granularity)
277.     {
278.         try
279.         {
280.             sleep(300);
281.         }
282.         catch (Exception e)
283.         {
284.         }
285.         ;
286.     }
287.     else
288.     {
289.         suspend();
290.     }
291. }
292. }
293.
294.@SuppressWarnings("deprecation")
295.class Animation extends Panel
296.{
297.     private static final long serialVersionUID = 1L;
298.     int dX = 30; // starting position for x coordinate
299.     int dBar = 15; // width of bar
300.     int dUnit = 15; // unit height
301.     int dDis = 20; // distance between bars
302.     Image offImage; // Offscreen Image
303.     Graphics offG; // Offscreen Graphics
304.     Font font = new Font("TimesRoman", Font.PLAIN, 14);
305.
306.     public Animation()
307.     {
308.         repaint();
309.     }
310.
311.     public Dimension preferredSize()
312.     {
313.         return new Dimension(320, 300);
314.     }
315.
316.     public Dimension minimumSize()
317.     {

```

```

318.     return preferredSize();
319. }
320.
321. public void update(Graphics g)
322. {
323.     paint(g);
324. }
325.
326. public void paint(Graphics g)
327. {
328.     Dimension d = size();
329.     if (offImage == null)
330.     {
331.         offImage = createImage(d.width, d.height);
332.         offG = offImage.getGraphics();
333.         offG.setFont(font);
334.     }
335.     offG.setColor(Color.yellow); // Set background of Animation Canvas to be
336.     //yellow
337.     offG.fillRect(0, 0, d.width, d.height); // Draw background
338.     offG.setColor(Color.blue); // Set color for bars to be blue
339.     int x = 40; //x coordinate of the bar position
340.     for (int i = 0; i < SortingApplet.SourceData.length; i++, x = x + dDis
341.          + dBar)
342.     {
343.         if (((i == Algorithm.indexj) || (i == Algorithm.indexj + 1))
344.             && (Algorithm.flag == 1))
345.         {
346.             offG.setColor(Color.green); // Use green color to indicate the
347.             // bars being compared currently
348.         }
349.         else if (((i > Algorithm.indexi) && (Algorithm.flag == 1))
350.                  || ((Algorithm.indexi == 0) && (Algorithm.indexj == 1) && (Algorithm.flag == -1)))
351.         {
352.             offG.setColor(Color.black); // Use black color to indicate bars
353.             // that are already sorted
354.         }
355.         offG.fillRect(x, 180 - SortingApplet.SourceData[i] * dUnit, dBar,
356.                         SortingApplet.SourceData[i] * dUnit); //fill bars
357.         offG.setColor(Color.blue); //Reset color to be blue
358.         offG.drawString(" " + i, x + 7, 25); // Draw the current index value
359.         //of i
360.         offG.drawString(" " + SortingApplet.SourceData[i], x + 7, 40);
361.     }
362.     offG.drawString("Index:", 5, 25);
363.     offG.drawString("Value:", 5, 40);
364.     offG.drawString("I:", 5, 205);
365.     offG.drawString("J:", 5, 230);
366.     offG.drawString("J+1:", 5, 255);
367.     if (Algorithm.indexi != 0)
368.         offG.drawString("i", 40 + 9 + Algorithm.indexi * (dDis + dBar), 205);
369.     if (Algorithm.indexj < Algorithm.indexi)
370.     {
371.         offG.drawString("j", 40 + 9 + Algorithm.indexj * (dDis + dBar), 230);
372.         offG.drawString("j+1", 40 + 7 + (Algorithm.indexj + 1)
373.                         * (dDis + dBar), 255); // Mark the current j+1 position
374.     }
375.     g.drawImage(offImage, 0, 0, this);
376. }
377. }
378.
379.@SuppressWarnings("deprecation")
380.public class SortingApplet extends Applet
381.{
382.    private static final long serialVersionUID = 1L;
383.    // Source code for Bubble Sort Algorithm

```

```

384. String code[] = { " for ( int i = n - 1; i >= 1; i-- )      ",
385.           "   for ( int j = 0; j <= i - 1; j++ )  ",
386.           "     if ( data [j] > data[j+1] ) {   ",
387.           "       int temp = data [j];          ",
388.           "       data [j] = data [j+1];        ",
389.           "       data [j+1] = temp;          ",
390.           "     }                           " };
391. // Explanation for each line of code
392. String pseudoCode[] = {
393.     "go through elements of 'data' from last to 1 index",
394.     "go through elements of 'data' from 0 to i index",
395.     "to compare data [j] and data [j+1]",
396.     "before swap, remember data [j]", "assign data [j] = data [j+1]",
397.     "assign data [j+1] the original value of data [j]",
398.     "end of if statement" };
399. public static int SourceData[] = { 7, 4, 5, 1, 8, 3, 6, 2 };
400. public static int normalData[] = { 7, 4, 5, 1, 8, 3, 6, 2 };
401. public static int bestData[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
402. public static int worstData[] = { 8, 7, 6, 5, 4, 3, 2, 1 };
403. Button sortButton = new Button(" Sort "); // sort Button
404. Button stopButton = new Button(" Stop "); // stop Button
405. Button resetButton = new Button(" Reset "); // reset Button
406. int choice = 0; // choice index
407. Choice dataChoice = new Choice(); // choice of different data sets
408. String dataLabels[] = { "normal case", "best case", "worst case" };
409. int granularity = 0; // granularity index
410. Choice granularityChoice = new Choice(); // choice of different granularity
411. String granularityLabels[] = { "entire sort", "next swap", "next line" }; // granularity
412. // labels
413. private Panel controlPanel = new Panel();
414. CodePanel codeDisplay = new CodePanel(code, pseudoCode, this);
415. Algorithm algorithm = null;
416. Animation animation = new Animation();
417.
418. public void init()
419. {
420.     setLayout(new BorderLayout()); // Set the panel to be border layout
421.     add("West", codeDisplay);
422.     add("East", animation);
423.     add("South", controlPanel);
424.     controlPanel.setLayout(new FlowLayout(FlowLayout.CENTER)); // Set
425.                                         // codepanel
426.                                         // to be
427.                                         // flowlayout
428.     controlPanel.add(sortButton); // Add sort button
429.     controlPanel.add(stopButton); // Add stop button
430.     stopButton.disable(); // At the beginning, stop button is disabled
431.     controlPanel.add(resetButton); // Add reset button
432.     resetButton.disable(); // At the beginning, resuet button is disabled
433.     controlPanel.add(dataChoice); // Add data choice menu
434.     for (int i = 0; i < dataLabels.length; i++)
435.     {
436.         dataChoice.addItem(dataLabels[i]); // Set label for each menu items
437.     }
438.     controlPanel.add(granularityChoice); // Add granularity choice menu
439.     for (int i = 0; i < granularityLabels.length; i++)
440.     {
441.         granularityChoice.addItem(granularityLabels[i]); // Set label for
442.                                         // each menu items
443.     }
444. }
445.
446. public void finished()
447. {
448.     algorithm = null;
449. }

```

```

450.
451. public boolean action(Event event, Object what)
452. {
453.     if (event.target == sortButton)
454.     {
455.         if (granularity == 0)
456.             sortButton.disable();
457.         else
458.             sortButton.setLabel("Continue");
459.         resetButton.disable(); // Once sorting begins, reset Button is
460.         // disabled
461.         stopButton.enable(); // stop Button is enabled
462.         if (algorithm == null)
463.         {
464.             algorithm = new Algorithm(codeDisplay, granularity, this,
465.                                         animation);
466.             algorithm.start(); // Start the Bubble Sort Algorithm
467.         }
468.         else
469.         {
470.             algorithm.resume(); // Continue the Bubble Sort Algorithm
471.         }
472.     }
473.     else if (event.target == stopButton)
474.     { // stop Button is clicked
475.         algorithm.stop(); // Stop the Bubble Sort Algorithm
476.         sortButton.disable(); // Disable the sort Button
477.         stopButton.disable(); // Disable the stop Button
478.         resetButton.enable(); // Enable the reset Button
479.         finished(); // Applet finished
480.     }
481.     else if (event.target == resetButton)
482.     { // reset Button is clicked
483.         finished(); // Applet finished
484.         sortButton.setLabel(" Sort "); // Set sort Button label
485.         sortButton.enable();
486.         stopButton.disable();
487.         resetdataArray(); // Recover the data array to its initial value
488.         // based on the dataChoice menu
489.         Algorithm.flag = -1; // Reset flag to initial value
490.         animation.repaint(); // Refresh the animation canvas
491.     }
492.     else if (event.target == dataChoice)
493.     { // If dataChoice menu is changed
494.         choice = dataChoice.getSelectedIndex();
495.     }
496.     else if (event.target == granularityChoice)
497.     { // If granularityChoice menu is changed
498.         granularity = granularityChoice.getSelectedIndex();
499.         if (algorithm != null)
500.         {
501.             algorithm.setGranularity(granularity); // Set the granularity to
502.             // be the new value in
503.             // the menu
504.         }
505.     }
506.     else
507.     {
508.         return false;
509.     }
510.     return true;
511. }
512.
513. public void resetdataArray()
514. {
515.     // Reset loop index to its initial value

```

```
516. Algorithm.indexi = 0;
517. Algorithm.indexj = 0;
518. if (choice == 0)
519. { // "Normal Case" is selected
520. // Reset the source data to be the normal case
521. for (int i = 0; i < normalData.length; i++)
522. {
523.     SourceData[i] = normalData[i];
524. }
525. }
526. else if (choice == 1)
527. { // "Best Case" is selected
528. // Reset the source data to be the best case
529. for (int i = 0; i < bestData.length; i++)
530. {
531.     SourceData[i] = bestData[i];
532. }
533. }
534. else if (choice == 2)
535. { // "Worst Case" is selected
536. // Reset the source data to be the worst case
537. for (int i = 0; i < worstData.length; i++)
538. {
539.     SourceData[i] = worstData[i];
540. }
541. }
542. }
543. }
```

Output:

advertisement

```
$ javac SortingApplet.java
$ java SortingApplet
```

Run this applet to see the output

308. Java Program to Find the Vertex Connectivity of a Graph

[« Prev](#)

[Next »](#)

This is a java program to find the vertex connectivity of a graph. Vertex connectivity simply means number of articulation points in a graph, articulation points are vertices of a graph whom removed makes graph disconnected.

Here is the source code of the Java Program to Find the Vertex Connectivity of a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. // Number of articulation points in a graph
3. package com.sanfoundry.graph;
4.
5. import java.util.Iterator;
6. import java.util.NoSuchElementException;
7. import java.util.Scanner;
8. import java.util.Stack;
9.
10.class VCBag<Item> implements Iterable<Item>
11.
12. private int N; // number of elements in VCBag
13. private Node<Item> first; // beginning of VCBag
14.
15. // helper linked list class
16. private static class Node<Item>
17.
18.     private Item item;
19.     private Node<Item> next;
20. }
21.
22. public VCBag()
23.
24.     first = null;
25.     N = 0;
26. }
27.
28. public boolean isEmpty()
29.
30.     return first == null;
31. }
32.
33. public int size()
34.
35.     return N;
36. }
37.
38. public void add(Item item)
39.
40.     Node<Item> oldfirst = first;
41.     first = new Node<Item>();
42.     first.item = item;
43.     first.next = oldfirst;
44.     N++;
45. }
46.
47. public Iterator<Item> iterator()
48.
49.     return new ListIterator<Item>(first);
50. }
51.
```

```

52. // an iterator, doesn't implement remove() since it's optional
53. @SuppressWarnings("hiding")
54. private class ListIterator<Item> implements Iterator<Item>
55. {
56.     private Node<Item> current;
57.
58.     public ListIterator(Node<Item> first)
59.     {
60.         current = first;
61.     }
62.
63.     public boolean hasNext()
64.     {
65.         return current != null;
66.     }
67.
68.     public void remove()
69.     {
70.         throw new UnsupportedOperationException();
71.     }
72.
73.     public Item next()
74.     {
75.         if (!hasNext())
76.             throw new NoSuchElementException();
77.         Item item = current.item;
78.         current = current.next;
79.         return item;
80.     }
81. }
82.
83.
84.class VCGraph
85.
86. private final int V;
87. private int E;
88. private VCBag<Integer>[] adj;
89.
90. @SuppressWarnings("unchecked")
91. public VCGraph(int V)
92. {
93.     if (V < 0)
94.         throw new IllegalArgumentException(
95.             "Number of vertices must be nonnegative");
96.     this.V = V;
97.     this.E = 0;
98.     adj = (VCBag<Integer>[]) new VCBag[V];
99.     for (int v = 0; v < V; v++)
100.    {
101.        adj[v] = new VCBag<Integer>();
102.    }
103.    System.out.println("Enter the number of edges: ");
104.    Scanner sc = new Scanner(System.in);
105.    int E = sc.nextInt();
106.    if (E < 0)
107.    {
108.        sc.close();
109.        throw new IllegalArgumentException(
110.            "Number of edges must be nonnegative");
111.    }
112.    System.out.println("Enter the edges: <from> <to>");
113.    for (int i = 0; i < E; i++)
114.    {
115.        int v = sc.nextInt();
116.        int w = sc.nextInt();
117.        addEdge(v, w);

```

```

118.     }
119.     sc.close();
120. }
121.
122. public VCGraph(VCGraph G)
123. {
124.     this(G.V());
125.     this.E = G.E();
126.     for (int v = 0; v < G.V(); v++)
127.     {
128.         // reverse so that adjacency list is in same order as original
129.         Stack<Integer> reverse = new Stack<Integer>();
130.         for (int w : G.adj[v])
131.         {
132.             reverse.push(w);
133.         }
134.         for (int w : reverse)
135.         {
136.             adj[v].add(w);
137.         }
138.     }
139. }
140.
141. public int V()
142. {
143.     return V;
144. }
145.
146. public int E()
147. {
148.     return E;
149. }
150.
151. public void addEdge(int v, int w)
152. {
153.     if (v < 0 || v >= V)
154.         throw new IndexOutOfBoundsException();
155.     if (w < 0 || w >= V)
156.         throw new IndexOutOfBoundsException();
157.     E++;
158.     adj[v].add(w);
159.     adj[w].add(v);
160. }
161.
162. public Iterable<Integer> adj(int v)
163. {
164.     if (v < 0 || v >= V)
165.         throw new IndexOutOfBoundsException();
166.     return adj[v];
167. }
168.
169. public String toString()
170. {
171.     StringBuilder s = new StringBuilder();
172.     String NEWLINE = System.getProperty("line.separator");
173.     s.append(V + " vertices, " + E + " edges " + NEWLINE);
174.     for (int v = 0; v < V; v++)
175.     {
176.         s.append(v + ": ");
177.         for (int w : adj[v])
178.         {
179.             s.append(w + " ");
180.         }
181.         s.append(NEWLINE);
182.     }
183.     return s.toString();

```

```

184. }
185. }
186.
187.public class VertexConnectivity
188. {
189.     private int[] low;
190.     private int[] pre;
191.     private int cnt;
192.     private boolean[] articulation;
193.
194.     public VertexConnectivity(VCGraph G)
195.     {
196.         low = new int[G.V()];
197.         pre = new int[G.V()];
198.         articulation = new boolean[G.V()];
199.         for (int v = 0; v < G.V(); v++)
200.             low[v] = -1;
201.         for (int v = 0; v < G.V(); v++)
202.             pre[v] = -1;
203.         for (int v = 0; v < G.V(); v++)
204.             if (pre[v] == -1)
205.                 dfs(G, v, v);
206.     }
207.
208.     private void dfs(VCGraph G, int u, int v)
209.     {
210.         int children = 0;
211.         pre[v] = cnt++;
212.         low[v] = pre[v];
213.         for (int w : G.adj(v))
214.         {
215.             if (pre[w] == -1)
216.             {
217.                 children++;
218.                 dfs(G, v, w);
219.                 // update low number
220.                 low[v] = Math.min(low[v], low[w]);
221.                 // non-root of DFS is an articulation point if low[w] >= pre[v]
222.                 if (low[w] >= pre[v] && u != v)
223.                     articulation[v] = true;
224.             }
225.             // update low number - ignore reverse of edge leading to v
226.             else if (w != u)
227.                 low[v] = Math.min(low[v], pre[w]);
228.         }
229.         // root of DFS is an articulation point if it has more than 1 child
230.         if (u == v && children > 1)
231.             articulation[v] = true;
232.     }
233.
234.     // is vertex v an articulation point?
235.     public boolean isArticulation(int v)
236.     {
237.         return articulation[v];
238.     }
239.
240.     // test client
241.     public static void main(String[] args)
242.     {
243.         Scanner sc = new Scanner(System.in);
244.         System.out.println("Enter the number of vertices: ");
245.         VCGraph G = new VCGraph(sc.nextInt());
246.         System.out.println(G);
247.         VertexConnectivity bic = new VertexConnectivity(G);
248.         int count = 0;
249.         for (int v = 0; v < G.V(); v++)

```

```
250.     if (bic.isArticulation(v))
251.         count++;
252.     System.out.println("Vertex Connectivity: " + count);
253.     sc.close();
254. }
255.}
```

Output:

advertisement

```
$ javac VertexConnectivity.java
$ java VertexConnectivity
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges: <from> <to>

0 1

1 2

1 3

3 4

4 5

5 3

5 2

6 vertices, 7 edges

0: 1

1: 3 2 0

2: 5 1

3: 5 4 1

4: 5 3

5: 2 3 4

Vertex Connectivity: 1

309. Java Program to Find the Edge Connectivity of a Graph

[« Prev](#)

[Next »](#)

This is a java program find the edge connectivity of the given graph. The edge connectivity simply means, number of bridges in a graph, bridges are edges when removed makes the graph disconnected.

Here is the source code of the Java Program to Find the Edge Connectivity of a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.  
2. // Number of bridges in a graph  
3. package com.sanfoundry.graph;  
4.  
5. import java.util.Iterator;  
6. import java.util.NoSuchElementException;  
7. import java.util.Scanner;  
8. import java.util.Stack;  
9.  
10. class ECBag<Item> implements Iterable<Item>  
11. {  
12.     private int      N; // number of elements in ECBag  
13.     private Node<Item> first; // beginning of ECBag  
14.  
15.     // helper linked list class  
16.     private static class Node<Item>  
17.     {  
18.         private Item      item;  
19.         private Node<Item> next;  
20.     }  
21.  
22.     public ECBag()  
23.     {  
24.         first = null;  
25.         N = 0;  
26.     }  
27.  
28.     public boolean isEmpty()  
29.     {  
30.         return first == null;  
31.     }  
32.  
33.     public int size()  
34.     {  
35.         return N;  
36.     }  
37.  
38.     public void add(Item item)  
39.     {  
40.         Node<Item> oldfirst = first;  
41.         first = new Node<Item>();  
42.         first.item = item;  
43.         first.next = oldfirst;  
44.         N++;  
45.     }  
46.  
47.     public Iterator<Item> iterator()  
48.     {  
49.         return new ListIterator<Item>(first);  
50.     }  
51.
```

```

52. // an iterator, doesn't implement remove() since it's optional
53. @SuppressWarnings("hiding")
54. private class ListIterator<Item> implements Iterator<Item>
55. {
56.     private Node<Item> current;
57.
58.     public ListIterator(Node<Item> first)
59.     {
60.         current = first;
61.     }
62.
63.     public boolean hasNext()
64.     {
65.         return current != null;
66.     }
67.
68.     public void remove()
69.     {
70.         throw new UnsupportedOperationException();
71.     }
72.
73.     public Item next()
74.     {
75.         if (!hasNext())
76.             throw new NoSuchElementException();
77.         Item item = current.item;
78.         current = current.next;
79.         return item;
80.     }
81. }
82.
83.
84.class BridgeGraph
85.
86. private final int V;
87. private int E;
88. private ECBag<Integer>[] adj;
89.
90. @SuppressWarnings("unchecked")
91. public BridgeGraph(int V)
92. {
93.     if (V < 0)
94.         throw new IllegalArgumentException(
95.             "Number of vertices must be nonnegative");
96.     this.V = V;
97.     this.E = 0;
98.     adj = (ECBag<Integer>[]) new ECBag[V];
99.     for (int v = 0; v < V; v++)
100.    {
101.        adj[v] = new ECBag<Integer>();
102.    }
103.    System.out.println("Enter the number of edges: ");
104.    Scanner sc = new Scanner(System.in);
105.    int E = sc.nextInt();
106.    if (E < 0)
107.    {
108.        sc.close();
109.        throw new IllegalArgumentException(
110.            "Number of edges must be nonnegative");
111.    }
112.    for (int i = 0; i < E; i++)
113.    {
114.        int v = sc.nextInt();
115.        int w = sc.nextInt();
116.        addEdge(v, w);
117.    }

```

```

118.     sc.close();
119. }
120.
121. public BridgeGraph(BridgeGraph G)
122. {
123.     this(G.V());
124.     this.E = G.E();
125.     for (int v = 0; v < G.V(); v++)
126.     {
127.         // reverse so that adjacency list is in same order as original
128.         Stack<Integer> reverse = new Stack<Integer>();
129.         for (int w : G.adj[v])
130.         {
131.             reverse.push(w);
132.         }
133.         for (int w : reverse)
134.         {
135.             adj[v].add(w);
136.         }
137.     }
138. }
139.
140. public int V()
141. {
142.     return V;
143. }
144.
145. public int E()
146. {
147.     return E;
148. }
149.
150. public void addEdge(int v, int w)
151. {
152.     if (v < 0 || v >= V)
153.         throw new IndexOutOfBoundsException();
154.     if (w < 0 || w >= V)
155.         throw new IndexOutOfBoundsException();
156.     E++;
157.     adj[v].add(w);
158.     adj[w].add(v);
159. }
160.
161. public Iterable<Integer> adj(int v)
162. {
163.     if (v < 0 || v >= V)
164.         throw new IndexOutOfBoundsException();
165.     return adj[v];
166. }
167.
168. public String toString()
169. {
170.     StringBuilder s = new StringBuilder();
171.     String NEWLINE = System.getProperty("line.separator");
172.     s.append(V + " vertices, " + E + " edges " + NEWLINE);
173.     for (int v = 0; v < V; v++)
174.     {
175.         s.append(v + ": ");
176.         for (int w : adj[v])
177.         {
178.             s.append(w + " ");
179.         }
180.         s.append(NEWLINE);
181.     }
182.     return s.toString();
183. }

```

```

184. }
185.
186.public class EdgeConnectivity
187. {
188.   private int bridges; // number of bridges
189.   private int cnt; // counter
190.   private int[] pre; // pre[v] = order in which dfs examines v
191.   private int[] low; // low[v] = lowest preorder of any vertex connected
192.           // to v
193.
194.   public EdgeConnectivity(BridgeGraph G)
195.   {
196.     low = new int[G.V0];
197.     pre = new int[G.V0];
198.     for (int v = 0; v < G.V0; v++)
199.       low[v] = -1;
200.     for (int v = 0; v < G.V0; v++)
201.       pre[v] = -1;
202.     for (int v = 0; v < G.V0; v++)
203.       if (pre[v] == -1)
204.         dfs(G, v, v);
205.   }
206.
207.   public int components()
208.   {
209.     return bridges + 1;
210.   }
211.
212.   private void dfs(BridgeGraph G, int u, int v)
213.   {
214.     pre[v] = cnt++;
215.     low[v] = pre[v];
216.     for (int w : G.adj(v))
217.     {
218.       if (pre[w] == -1)
219.       {
220.         dfs(G, v, w);
221.         low[v] = Math.min(low[v], low[w]);
222.         if (low[w] == pre[w])
223.         {
224.           // System.out.println(v + "-" + w + " is a bridge");
225.           bridges++;
226.         }
227.       }
228.     } // update low number - ignore reverse of edge leading to v
229.     else if (w != u)
230.       low[v] = Math.min(low[v], pre[w]);
231.   }
232. }
233.
234. public static void main(String[] args)
235. {
236.   Scanner sc = new Scanner(System.in);
237.   System.out.println("Enter the number of vertices: ");
238.   BridgeGraph G = new BridgeGraph(sc.nextInt());
239.   System.out.println(G);
240.   EdgeConnectivity bridge = new EdgeConnectivity(G);
241.   System.out.println("Edge Connectivity: " + bridge.bridges);
242.   sc.close();
243. }
244. }
```

Output:

advertisement

\$ javac EdgeConnectivity.java

```
$ java EdgeConnectivity
```

```
Enter the number of vertices:
```

```
6
```

```
Enter the number of edges:
```

```
7
```

```
0 1
```

```
1 2
```

```
1 3
```

```
3 4
```

```
4 5
```

```
5 3
```

```
5 2
```

```
6 vertices, 7 edges
```

```
0: 1
```

```
1: 3 2 0
```

```
2: 5 1
```

```
3: 5 4 1
```

```
4: 5 3
```

```
5: 2 3 4
```

```
Edge Connectivity: 1
```

310. Java Program to Implement an Algorithm to Find the Global min Cut in a Graph

[« Prev](#)

[Next »](#)

This is a java program to find global min cut of the graph. In computer science and graph theory, Karger's algorithm is a randomized algorithm to compute a minimum cut of a connected graph.

Here is the source code of the Java Program to Implement an Algorithm to Find the Global min Cut in a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.io.BufferedReader;
5. import java.io.FileReader;
6. import java.util.ArrayList;
7. import java.util.Comparator;
8. import java.util.HashSet;
9. import java.util.Iterator;
10.import java.util.LinkedHashMap;
11.import java.util.List;
12.import java.util.Map;
13.import java.util.Random;
14.import java.util.Set;
15.import java.util.TreeMap;
16.
17.public class GlobalMinCut
18.
19.    private static class Graph
20.    {
21.        private final Map<Integer, Vertex> vertices = new TreeMap<Integer, Vertex>(
22.            new Comparator<Integer>()
23.            {
24.                @Override
25.                public int compare(Integer arg0, Integer arg1)
26.                {
27.                    return arg0.compareTo(arg1);
28.                }
29.            });
30.        private final List<Edge> edges = new ArrayList<Edge>();
31.
32.        public void addVertex(Vertex v)
33.        {
34.            vertices.put(v.lbl, v);
35.        }
36.
37.        public Vertex getVertex(int lbl)
38.        {
39.            Vertex v;
40.            if ((v = vertices.get(lbl)) == null)
41.            {
42.                v = new Vertex(lbl);
43.                addVertex(v);
44.            }
45.            return v;
46.        }
47.    }
48.
49.    private static class Vertex
50.    {
```

```

51. private final int lbl;
52. private final Set<Edge> edges = new HashSet<Edge>();
53.
54. public Vertex(int lbl)
55. {
56.     this.lbl = lbl;
57. }
58.
59. public void addEdge(Edge edge)
60. {
61.     edges.add(edge);
62. }
63.
64. public Edge getEdgeTo(Vertex v2)
65. {
66.     for (Edge edge : edges)
67.     {
68.         if (edge.contains(this, v2))
69.             return edge;
70.     }
71.     return null;
72. }
73. }
74.
75. private static class Edge
76. {
77.     private final List<Vertex> ends = new ArrayList<Vertex>();
78.
79.     public Edge(Vertex fst, Vertex snd)
80.     {
81.         if (fst == null || snd == null)
82.         {
83.             throw new IllegalArgumentException("Both vertices are required");
84.         }
85.         ends.add(fst);
86.         ends.add(snd);
87.     }
88.
89.     public boolean contains(Vertex v1, Vertex v2)
90.     {
91.         return ends.contains(v1) && ends.contains(v2);
92.     }
93.
94.     public Vertex getOppositeVertex(Vertex v)
95.     {
96.         if (!ends.contains(v))
97.         {
98.             throw new IllegalArgumentException("Vertex " + v.lbl);
99.         }
100.        return ends.get(1 - ends.indexOf(v));
101.    }
102.
103.    public void replaceVertex(Vertex oldV, Vertex newV)
104.    {
105.        if (!ends.contains(oldV))
106.        {
107.            throw new IllegalArgumentException("Vertex " + oldV.lbl);
108.        }
109.        ends.remove(oldV);
110.        ends.add(newV);
111.    }
112. }
113.
114. public static int minCut(Graph gr)
115. {
116.     Random rnd = new Random();

```

```

117.     while (gr.vertices.size() > 2)
118.     {
119.         Edge edge = gr.edges.remove(rnd.nextInt(gr.edges.size()));
120.         Vertex v1 = cleanVertex(gr, edge.ends.get(0), edge);
121.         Vertex v2 = cleanVertex(gr, edge.ends.get(1), edge);
122.         // contract
123.         Vertex mergedVertex = new Vertex(v1.lbl);
124.         redirectEdges(gr, v1, mergedVertex);
125.         redirectEdges(gr, v2, mergedVertex);
126.         gr.addVertex(mergedVertex);
127.     }
128.     return gr.edges.size();
129. }
130.
131. private static Vertex cleanVertex(Graph gr, Vertex v, Edge e)
132. {
133.     gr.vertices.remove(v.lbl);
134.     v.edges.remove(e);
135.     return v;
136. }
137.
138. private static void redirectEdges(Graph gr, Vertex fromV, Vertex toV)
139. {
140.     for (Iterator<Edge> it = fromV.edges.iterator(); it.hasNext());
141.     {
142.         Edge edge = it.next();
143.         it.remove();
144.         if (edge.getOppositeVertex(fromV) == toV)
145.         {
146.             // remove self-loop
147.             toV.edges.remove(edge);
148.             gr.edges.remove(edge);
149.         }
150.         else
151.         {
152.             edge.replaceVertex(fromV, toV);
153.             toV.addEdge(edge);
154.         }
155.     }
156. }
157.
158. public static int[][] getArray(String relPath)
159. {
160.     Map<Integer, List<Integer>> vertices = new LinkedHashMap<Integer, List<Integer>>();
161.     FileReader fr;
162.     try
163.     {
164.         fr = new FileReader(relPath);
165.         BufferedReader br = new BufferedReader(fr);
166.         String line;
167.         while ((line = br.readLine()) != null)
168.         {
169.             String[] split = line.trim().split("(\\s)+");
170.             List<Integer> adjList = new ArrayList<Integer>();
171.             for (int i = 1; i < split.length; i++)
172.             {
173.                 adjList.add(Integer.parseInt(split[i]) - 1);
174.             }
175.             vertices.put(Integer.parseInt(split[0]) - 1, adjList);
176.         }
177.         fr.close();
178.     }
179.     catch (Exception e)
180.     {
181.         e.printStackTrace();
182.     }

```

```

183.     int[][] array = new int[vertices.size()][];
184.     for (Map.Entry<Integer, List<Integer>> entry : vertices.entrySet())
185.     {
186.         List<Integer> adjList = entry.getValue();
187.         int[] adj = new int[adjList.size()];
188.         for (int i = 0; i < adj.length; i++)
189.         {
190.             adj[i] = adjList.get(i);
191.         }
192.         array[entry.getKey()] = adj;
193.     }
194.     return array;
195. }
196.
197. private static Graph createGraph(int[][] array)
198. {
199.     Graph gr = new Graph();
200.     for (int i = 0; i < array.length; i++)
201.     {
202.         Vertex v = gr.getVertex(i);
203.         for (int edgeTo : array[i])
204.         {
205.             Vertex v2 = gr.getVertex(edgeTo);
206.             Edge e;
207.             if ((e = v2.getEdgeTo(v)) == null)
208.             {
209.                 e = new Edge(v, v2);
210.                 gr.edges.add(e);
211.                 v.addEdge(e);
212.                 v2.addEdge(e);
213.             }
214.         }
215.     }
216.     return gr;
217. }
218.
219. /**
220. * @param args
221. */
222. public static void main(String[] args)
223. {
224.     int[][] arr = getArray("GlobalMinCut.txt");
225.     Map<Integer, Integer> statistics = new LinkedHashMap<Integer, Integer>();
226.     int min = arr.length;
227.     int iter = arr.length * arr.length;
228.     Graph g = createGraph(arr);
229.     printGraph(g);
230.     for (int i = 0; i < iter; i++)
231.     {
232.         Graph gr = createGraph(arr);
233.         int currMin = minCut(gr);
234.         min = Math.min(min, currMin);
235.         Integer counter;
236.         if ((counter = statistics.get(currMin)) == null)
237.         {
238.             counter = 0;
239.         }
240.         statistics.put(currMin, counter + 1);
241.     }
242.     System.out.println("Min: " + min + " stat: "
243.                         + (statistics.get(min) * 100 / iter) + "%");
244. }
245.
246. private static void printGraph(Graph gr)
247. {
248.     System.out.println("Printing graph");

```

```
249.     for (Vertex v : gr.vertices.values())
250.     {
251.         System.out.print(v.lbl + ":");
252.         for (Edge edge : v.edges)
253.         {
254.             System.out.print(" " + edge.getOppositeVertex(v).lbl);
255.         }
256.         System.out.println();
257.     }
258. }
259. }
```

Output:

advertisement

```
$ javac GlobalMinCut.java
$ java GlobalMinCut
```

```
Printing graph
0: 35 38 17 18 14 22
1: 35 8 17 3 25 22
2: 34 15 5 10
3: 23 1 17 22
4: 7 20 13 28
5: 2 33 15 34
6: 32 27 37 29
7: 13 11 30 4 28
8: 38 16 12 1 9 19
9: 28 8 11 13 19
10: 15 32 2 25 29
11: 19 13 7 9
12: 8 23 38 19
13: 11 7 9 4
14: 18 35 25 0
15: 34 31 2 10 29 16 5
16: 8 15 39 37 27 31
17: 0 38 23 1 3
18: 14 25 0 26
19: 11 12 8 9
20: 28 4 24 36
21: 31 33 39 34
22: 35 0 1 3
23: 3 17 12 38
24: 30 28 36 20
25: 26 18 14 10 1 30
26: 25 36 28 30 18
27: 31 6 37 16
28: 9 26 20 24 7 4
29: 36 15 32 6 10
30: 7 24 26 25 36
31: 15 21 27 39 16
32: 6 10 29 37
33: 21 5 39 34
34: 15 2 21 5 33
35: 0 1 14 22
36: 29 26 24 30 20
37: 39 16 6 27 32
38: 0 8 17 12 23
39: 37 31 21 16 33
Min: 3 stat: 6%
```

311. Java Program to Check Whether a Weak Link i.e. Articulation Vertex Exists in a Graph or Check Whether G is Biconnected or Not

[« Prev](#)

[Next »](#)

This is a java program check if the graph contains any weak link (articulation point). A vertex in an undirected connected graph is an articulation point (or cut vertex) iff removing it (and edges through it) disconnects the graph. Articulation points represent vulnerabilities in a connected network – single points whose failure would split the network into 2 or more disconnected components. They are useful for designing reliable networks.

Here is the source code of the Java Program to Check Whether a Weak Link i.e. Articulation Vertex Exists in a Graph or Check Whether G is Biconnected or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.hinguapps.graph;
3.
4. import java.util.Iterator;
5. import java.util.NoSuchElementException;
6. import java.util.Scanner;
7. import java.util.Stack;
8.
9. class Bag<Item> implements Iterable<Item>
10. {
11.     private int      N; // number of elements in bag
12.     private Node<Item> first; // beginning of bag
13.
14.     // helper linked list class
15.     private static class Node<Item>
16.     {
17.         private Item      item;
18.         private Node<Item> next;
19.     }
20.
21.     public Bag()
22.     {
23.         first = null;
24.         N = 0;
25.     }
26.
27.     public boolean isEmpty()
28.     {
29.         return first == null;
30.     }
31.
32.     public int size()
33.     {
34.         return N;
35.     }
36.
37.     public void add(Item item)
38.     {
39.         Node<Item> oldfirst = first;
40.         first = new Node<Item>();
41.         first.item = item;
42.         first.next = oldfirst;
43.         N++;
44.     }
45.
46.     public Iterator<Item> iterator()
```

```

47. {
48.     return new ListIterator<Item>(first);
49. }
50.
51. // an iterator, doesn't implement remove() since it's optional
52. private class ListIterator<Item> implements Iterator<Item>
53. {
54.     private Node<Item> current;
55.
56.     public ListIterator(Node<Item> first)
57.     {
58.         current = first;
59.     }
60.
61.     public boolean hasNext()
62.     {
63.         return current != null;
64.     }
65.
66.     public void remove()
67.     {
68.         throw new UnsupportedOperationException();
69.     }
70.
71.     public Item next()
72.     {
73.         if (!hasNext())
74.             throw new NoSuchElementException();
75.         Item item = current.item;
76.         current = current.next;
77.         return item;
78.     }
79. }
80. }
81.
82.class APGraph
83.{
84.    private final int    V;
85.    private int          E;
86.    private Bag<Integer>[] adj;
87.
88.    public APGraph(int V)
89.    {
90.        if (V < 0)
91.            throw new IllegalArgumentException(
92.                "Number of vertices must be nonnegative");
93.        this.V = V;
94.        this.E = 0;
95.        adj = (Bag<Integer>[]) new Bag[V];
96.        for (int v = 0; v < V; v++)
97.        {
98.            adj[v] = new Bag<Integer>();
99.        }
100.       System.out.println("Enter the number of edges: ");
101.       Scanner sc = new Scanner(System.in);
102.       int E = sc.nextInt();
103.       if (E < 0)
104.       {
105.           sc.close();
106.           throw new IllegalArgumentException(
107.               "Number of edges must be nonnegative");
108.       }
109.       for (int i = 0; i < E; i++)
110.       {
111.           int v = sc.nextInt();
112.           int w = sc.nextInt();

```

```

113.         addEdge(v, w);
114.     }
115.     sc.close();
116. }
117.
118. public APGraph(APGraph G)
119. {
120.     this(G.V());
121.     this.E = G.E();
122.     for (int v = 0; v < G.V(); v++)
123.     {
124.         // reverse so that adjacency list is in same order as original
125.         Stack<Integer> reverse = new Stack<Integer>();
126.         for (int w : G.adj[v])
127.         {
128.             reverse.push(w);
129.         }
130.         for (int w : reverse)
131.         {
132.             adj[v].add(w);
133.         }
134.     }
135. }
136.
137. public int V()
138. {
139.     return V;
140. }
141.
142. public int E()
143. {
144.     return E;
145. }
146.
147. public void addEdge(int v, int w)
148. {
149.     if (v < 0 || v >= V)
150.         throw new IndexOutOfBoundsException();
151.     if (w < 0 || w >= V)
152.         throw new IndexOutOfBoundsException();
153.     E++;
154.     adj[v].add(w);
155.     adj[w].add(v);
156. }
157.
158. public Iterable<Integer> adj(int v)
159. {
160.     if (v < 0 || v >= V)
161.         throw new IndexOutOfBoundsException();
162.     return adj[v];
163. }
164.
165. public String toString()
166. {
167.     StringBuilder s = new StringBuilder();
168.     String NEWLINE = System.getProperty("line.separator");
169.     s.append(V + " vertices, " + E + " edges " + NEWLINE);
170.     for (int v = 0; v < V; v++)
171.     {
172.         s.append(v + ": ");
173.         for (int w : adj[v])
174.         {
175.             s.append(w + " ");
176.         }
177.         s.append(NEWLINE);
178.     }
}

```

```

179.     return s.toString();
180. }
181. }
182.
183.public class ArticulationPoints
184. {
185.     private int[] low;
186.     private int[] pre;
187.     private int cnt;
188.     private boolean[] articulation;
189.
190.     public ArticulationPoints(APGraph G)
191.     {
192.         low = new int[G.V()];
193.         pre = new int[G.V()];
194.         articulation = new boolean[G.V()];
195.         for (int v = 0; v < G.V(); v++)
196.             low[v] = -1;
197.         for (int v = 0; v < G.V(); v++)
198.             pre[v] = -1;
199.         for (int v = 0; v < G.V(); v++)
200.             if (pre[v] == -1)
201.                 dfs(G, v, v);
202.     }
203.
204.     private void dfs(APGraph G, int u, int v)
205.     {
206.         int children = 0;
207.         pre[v] = cnt++;
208.         low[v] = pre[v];
209.         for (int w : G.adj(v))
210.         {
211.             if (pre[w] == -1)
212.             {
213.                 children++;
214.                 dfs(G, v, w);
215.                 // update low number
216.                 low[v] = Math.min(low[v], low[w]);
217.                 // non-root of DFS is an articulation point if low[w] >= pre[v]
218.                 if (low[w] >= pre[v] && u != v)
219.                     articulation[v] = true;
220.             }
221.             // update low number - ignore reverse of edge leading to v
222.             else if (w != u)
223.                 low[v] = Math.min(low[v], pre[w]);
224.         }
225.         // root of DFS is an articulation point if it has more than 1 child
226.         if (u == v && children > 1)
227.             articulation[v] = true;
228.     }
229.
230.     // is vertex v an articulation point?
231.     public boolean isArticulation(int v)
232.     {
233.         return articulation[v];
234.     }
235.
236.     // test client
237.     public static void main(String[] args)
238.     {
239.         Scanner sc = new Scanner(System.in);
240.         System.out.println("Enter the number of vertices: ");
241.         APGraph G = new APGraph(sc.nextInt());
242.         System.out.println(G);
243.         ArticulationPoints bic = new ArticulationPoints(G);
244.         System.out.println("Articulation points: ");

```

```
245.     for (int v = 0; v < G.V(); v++)
246.         if (bic.isArticulation(v))
247.             System.out.println(v);
248.     sc.close();
249. }
250.}
```

Output:

advertisement

```
$ javac ArticulationPoints.java
$ java ArticulationPoints
```

Enter the number of vertices:

6

Enter the number of edges:

7

0 1

1 2

1 3

3 4

4 5

5 3

5 2

6 vertices, 7 edges

0: 1

1: 3 2 0

2: 5 1

3: 5 4 1

4: 5 3

5: 2 3 4

Articulation points:

1

312. Java Program to Check Whether it is Weakly Connected or Strongly Connected for a Directed Graph

[« Prev](#)

[Next »](#)

This is a java program to check whether a graph is strongly connected or weakly connected. If graph has more than one connected components it is weakly connected. If graph has only one connected component it is strongly connected.

Here is the source code of the Java Program to Check Whether it is Weakly Connected or Strongly Connected for a Directed Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.ArrayList;
5. import java.util.Iterator;
6. import java.util.List;
7. import java.util.Scanner;
8. import java.util.Stack;
9.
10. public class StronglyorWeaklyConnectedDigraphs
11. {
12.     private int          V;
13.     private int          preCount;
14.     private int[]         low;
15.     private boolean[]    visited;
16.     private List<Integer>[] graph;
17.     private List<List<Integer>> sccComp;
18.     private Stack<Integer> stack;
19.
20.     /** function to get all strongly connected components */
21.     public List<List<Integer>> getSCComponents(List<Integer>[] graph)
22.     {
23.         V = graph.length;
24.         this.graph = graph;
25.         low = new int[V];
26.         visited = new boolean[V];
27.         stack = new Stack<Integer>();
28.         sccComp = new ArrayList<>();
29.         for (int v = 0; v < V; v++)
30.             if (!visited[v])
31.                 dfs(v);
32.         return sccComp;
33.     }
34.
35.     /** function dfs */
36.     public void dfs(int v)
37.     {
38.         low[v] = preCount++;
39.         visited[v] = true;
40.         stack.push(v);
41.         int min = low[v];
42.         for (int w : graph[v])
43.         {
44.             if (!visited[w])
45.                 dfs(w);
46.             if (low[w] < min)
47.                 min = low[w];
```

```

48.    }
49.    if (min < low[v])
50.    {
51.        low[v] = min;
52.        return;
53.    }
54.    List<Integer> component = new ArrayList<Integer>();
55.    int w;
56.    do
57.    {
58.        w = stack.pop();
59.        component.add(w);
60.        low[w] = V;
61.    }
62.    while (w != v);
63.    sccComp.add(component);
64. }
65.
66. @SuppressWarnings("unchecked")
67. public static void main(String[] args)
68. {
69.     Scanner scan = new Scanner(System.in);
70.     System.out.println("Enter number of Vertices");
71.     /** number of vertices **/
72.     int V = scan.nextInt();
73.     /** make graph **/
74.     List<Integer>[] g = new List[V];
75.     for (int i = 0; i < V; i++)
76.         g[i] = new ArrayList<Integer>();
77.     /** accept all edges **/
78.     System.out.println("Enter number of edges");
79.     int E = scan.nextInt();
80.     /** all edges **/
81.     System.out.println("Enter the edges in the graph : <from> <to>");
82.     for (int i = 0; i < E; i++)
83.     {
84.         int x = scan.nextInt();
85.         int y = scan.nextInt();
86.         g[x].add(y);
87.     }
88.     StronglyConnectedGraph t = new StronglyConnectedGraph();
89.     System.out.print("The graph is : ");
90.     /** print all strongly connected components **/
91.     List<List<Integer>> scComponents = t.getSCComponents(g);
92.     Iterator<List<Integer>> iterator = scComponents.iterator();
93.     boolean weaklyConnected = false;
94.     while (iterator.hasNext())
95.     {
96.         if (iterator.next().size() == 1)
97.         {
98.             weaklyConnected = true;
99.         }
100.    }
101.    if (weaklyConnected == true)
102.        System.out.println("Weakly Connected");
103.    else
104.        System.out.println("Strongly Connected");
105.    scan.close();
106. }
107. }
```

Output:

advertisement

```
$ javac StronglyorWeaklyConnectedDigraphs.java
$ java StronglyorWeaklyConnectedDigraphs
```

Enter number of Vertices

6

Enter number of edges

7

Enter the edges in the graph : <from> <to>

0 1

1 2

1 3

3 4

4 5

5 3

5 2

The graph is : Weakly Connected

313. Java Program to Give an Implementation of the Traditional Chinese Postman Problem

[« Prev](#)

[Next »](#)

This is a java program to implement chinese Postman Problem. In graph theory, a branch of mathematics, the Chinese postman problem (CPP), postman tour or route inspection problem is to find a shortest closed path or circuit that visits every edge of a (connected) undirected graph. When the graph has an Eulerian circuit (a closed walk that covers every edge once), that circuit is an optimal solution. Otherwise, the optimization problem is to find the fewest number of edges to add to the graph so that the resulting multigraph does have an Eulerian circuit.

Here is the source code of the Java Program to Give an Implementation of the Traditional Chinese Postman Problem. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.Vector;
5.
6. public class ChinesePostmanProblem
7. {
8.     int N;           // number of vertices
9.     int delta[];    // deltas of vertices
10.    int neg[], pos[]; // unbalanced vertices
11.    int arcs[][];   // adjacency matrix, counts arcs between
12.        // vertices
13.    Vector<String> label[][]; // vectors of labels of arcs (for each
14.        // vertex
15.        // pair)
16.    int f[][];       // repeated arcs in CPT
17.    float c[][];    // costs of cheapest arcs or paths
18.    String cheapestLabel[][]; // labels of cheapest arcs
19.    boolean defined[][]; // whether path cost is defined between
20.        // vertices
21.    int path[][];   // spanning tree of the graph
22.    float basicCost; // total cost of traversing each arc once
23.
24.    void solve()
25.    {
26.        leastCostPaths();
27.        checkValid();
28.        findUnbalanced();
29.        findFeasible();
30.        while (improvements())
31.            ;
32.    }
33.
34.    // allocate array memory, and instantiate graph object
35.    @SuppressWarnings("unchecked")
36.    ChinesePostmanProblem(int vertices)
37.    {
38.        if ((N = vertices) <= 0)
39.            throw new Error("Graph is empty");
40.        delta = new int[N];
41.        defined = new boolean[N][N];
42.        label = new Vector[N][N];
43.        c = new float[N][N];
44.        f = new int[N][N];
45.        arcs = new int[N][N];
46.        cheapestLabel = new String[N][N];
47.        path = new int[N][N];
```

```

48.     basicCost = 0;
49. }
50.
51. ChinesePostmanProblem addArc(String lab, int u, int v, float cost)
52. {
53.     if (!defined[u][v])
54.         label[u][v] = new Vector<String>();
55.     label[u][v].addElement(lab);
56.     basicCost += cost;
57.     if (!defined[u][v] || c[u][v] > cost)
58.     {
59.         c[u][v] = cost;
60.         cheapestLabel[u][v] = lab;
61.         defined[u][v] = true;
62.         path[u][v] = v;
63.     }
64.     arcs[u][v]++;
65.     delta[u]++;
66.     delta[v]--;
67.     return this;
68. }
69.
70. void leastCostPaths()
71. {
72.     for (int k = 0; k < N; k++)
73.         for (int i = 0; i < N; i++)
74.             if (defined[i][k])
75.                 for (int j = 0; j < N; j++)
76.                     if (defined[k][j])
77.                         && (!defined[i][j] || c[i][j] > c[i][k]
78.                               + c[k][j]))
79.                     {
80.                         path[i][j] = path[i][k];
81.                         c[i][j] = c[i][k] + c[k][j];
82.                         defined[i][j] = true;
83.                         if (i == j && c[i][j] < 0)
84.                             return; // stop on negative cycle
85.                     }
86.     }
87.
88. void checkValid()
89. {
90.     for (int i = 0; i < N; i++)
91.     {
92.         for (int j = 0; j < N; j++)
93.             if (!defined[i][j])
94.                 throw new Error("Graph is not strongly connected");
95.             if (c[i][i] < 0)
96.                 throw new Error("Graph has a negative cycle");
97.     }
98. }
99.
100. float cost()
101. {
102.     return basicCost + phi();
103. }
104.
105. float phi()
106. {
107.     float phi = 0;
108.     for (int i = 0; i < N; i++)
109.         for (int j = 0; j < N; j++)
110.             phi += c[i][j] * f[i][j];
111.     return phi;
112. }
113.

```

```

114. void findUnbalanced()
115. {
116.     int nn = 0, np = 0; // number of vertices of negative/positive delta
117.     for (int i = 0; i < N; i++)
118.         if (delta[i] < 0)
119.             nn++;
120.         else if (delta[i] > 0)
121.             np++;
122.     neg = new int[nn];
123.     pos = new int[np];
124.     nn = np = 0;
125.     for (int i = 0; i < N; i++)
126.         // initialise sets
127.         if (delta[i] < 0)
128.             neg[nn++] = i;
129.         else if (delta[i] > 0)
130.             pos[np++] = i;
131.     }
132.
133. void findFeasible()
134. { // delete next 3 lines to be faster, but non-reentrant
135.     int delta[] = new int[N];
136.     for (int i = 0; i < N; i++)
137.         delta[i] = this.delta[i];
138.     for (int u = 0; u < neg.length; u++)
139.     {
140.         int i = neg[u];
141.         for (int v = 0; v < pos.length; v++)
142.         {
143.             int j = pos[v];
144.             f[i][j] = -delta[i] < delta[j] ? -delta[i] : delta[j];
145.             delta[i] += f[i][j];
146.             delta[j] -= f[i][j];
147.         }
148.     }
149. }
150.
151. boolean improvements()
152. {
153.     ChinesePostmanProblem residual = new ChinesePostmanProblem(N);
154.     for (int u = 0; u < neg.length; u++)
155.     {
156.         int i = neg[u];
157.         for (int v = 0; v < pos.length; v++)
158.         {
159.             int j = pos[v];
160.             residual.addArc(null, i, j, c[i][j]);
161.             if (f[i][j] != 0)
162.                 residual.addArc(null, j, i, -c[i][j]);
163.         }
164.     }
165.     residual.leastCostPaths(); // find a negative cycle
166.     for (int i = 0; i < N; i++)
167.         if (residual.c[i][i] < 0) // cancel the cycle (if any)
168.         {
169.             int k = 0, u, v;
170.             boolean kunset = true;
171.             u = i;
172.             do // find k to cancel
173.             {
174.                 v = residual.path[u][i];
175.                 if (residual.c[u][v] < 0 && (kunset || k > f[v][u]))
176.                 {
177.                     k = f[v][u];
178.                     kunset = false;
179.                 }

```

```

180.     }
181.     while ((u == v) != i);
182.     u = i;
183.     do // cancel k along the cycle
184.     {
185.         v = residual.path[u][i];
186.         if (residual.c[u][v] < 0)
187.             f[v][u] -= k;
188.         else
189.             f[u][v] += k;
190.     }
191.     while ((u == v) != i);
192.     return true; // have another go
193. }
194. return false; // no improvements found
195. }
196.
197. static final int NONE = -1; // anything < 0
198.
199. int findPath(int from, int f[][])
200. {
201.     for (int i = 0; i < N; i++)
202.         if (f[from][i] > 0)
203.             return i;
204.     return NONE;
205. }
206.
207. void printCPT(int startVertex)
208. {
209.     int v = startVertex;
210.     // delete next 7 lines to be faster, but non-reentrant
211.     int arcs[][] = new int[N][N];
212.     int f[][] = new int[N][N];
213.     for (int i = 0; i < N; i++)
214.         for (int j = 0; j < N; j++)
215.         {
216.             arcs[i][j] = this.arcs[i][j];
217.             f[i][j] = this.f[i][j];
218.         }
219.     while (true)
220.     {
221.         int u = v;
222.         if ((v = findPath(u, f)) != NONE)
223.         {
224.             f[u][v]--;
225.             for (int p; u != v; u = p) // break down path into its arcs
226.             {
227.                 p = path[u][v];
228.                 System.out.println("Take arc " + cheapestLabel[u][p]
229.                     + " from " + u + " to " + p);
230.             }
231.         }
232.     else
233.     {
234.         int bridgeVertex = path[u][startVertex];
235.         if (arcs[u][bridgeVertex] == 0)
236.             break; // finished if bridge already used
237.         v = bridgeVertex;
238.         for (int i = 0; i < N; i++)
239.             // find an unused arc, using bridge last
240.             if (i != bridgeVertex && arcs[u][i] > 0)
241.             {
242.                 v = i;
243.                 break;
244.             }
245.         arcs[u][v]--;
246.     }
247. }
248.
```

```

246.     System.out.println("Take arc "
247.         + label[u][v].elementAt(arcs[u][v]) + " from " + u
248.         + " to " + v); // use each arc label in turn
249.     }
250. }
252.
253. static public void main(String args[])
254. {
255.     // create a graph of four vertices
256.     ChinesePostmanProblem G = new ChinesePostmanProblem(4);
257.     // add the arcs for the example graph
258.     G.addArc("a", 0, 1, 1).addArc("b", 0, 2, 1).addArc("c", 1, 2, 1)
259.         .addArc("d", 1, 3, 1).addArc("e", 2, 3, 1).addArc("f", 3, 0, 1);
260.     G.solve(); // find the CPT
261.     G.printCPT(0); // print it, starting from vertex 0
262.     System.out.println("Cost = " + G.cost());
263.     OpenCPP.test();
264. }
265.
266. // Print arcs and f
267. void debugarcf()
268. {
269.     for (int i = 0; i < N; i++)
270.     {
271.         System.out.print("f[" + i + "] = ");
272.         for (int j = 0; j < N; j++)
273.             System.out.print(f[i][j] + " ");
274.         System.out.print(" arcs[" + i + "] = ");
275.         for (int j = 0; j < N; j++)
276.             System.out.print(arcs[i][j] + " ");
277.         System.out.println();
278.     }
279. }
280.
281. // Print out most of the matrices: defined, path and f
282. void debug()
283. {
284.     for (int i = 0; i < N; i++)
285.     {
286.         System.out.print(i + " ");
287.         for (int j = 0; j < N; j++)
288.             System.out
289.                 .print(j + ":" + (defined[i][j] ? "T" : "F") + " "
290.                     + c[i][j] + " p=" + path[i][j] + " f="
291.                     + f[i][j] + "; ");
292.         System.out.println();
293.     }
294. }
295.
296. // Print out non zero f elements, and phi
297. void debugf()
298. {
299.     float sum = 0;
300.     for (int i = 0; i < N; i++)
301.     {
302.         boolean any = false;
303.         for (int j = 0; j < N; j++)
304.             if (f[i][j] != 0)
305.             {
306.                 any = true;
307.                 System.out.print("f(" + i + "," + j + ":" + label[i][j]
308.                     + ")=" + f[i][j] + "@" + c[i][j] + " ");
309.                 sum += f[i][j] * c[i][j];
310.             }
311.         if (any)

```

```

312.         System.out.println();
313.     }
314.     System.out.println("-->phi=" + sum);
315. }
316.
317. // Print out cost matrix.
318. void debug()
319. {
320.     for (int i = 0; i < N; i++)
321.     {
322.         boolean any = false;
323.         for (int j = 0; j < N; j++)
324.             if (c[i][j] != 0)
325.             {
326.                 any = true;
327.                 System.out.print("c(" + i + "," + j + ":" + label[i][j]
328.                         + ")=" + c[i][j] + " ");
329.             }
330.         if (any)
331.             System.out.println();
332.     }
333. }
334. }
335.
336.class OpenCPP
337.{
338.    class Arc
339.    {
340.        String lab;
341.        int u, v;
342.        float cost;
343.
344.        Arc(String lab, int u, int v, float cost)
345.        {
346.            this.lab = lab;
347.            this.u = u;
348.            this.v = v;
349.            this.cost = cost;
350.        }
351.    }
352.
353.    Vector<Arc> arcs = new Vector<Arc>();
354.    int N;
355.
356.    OpenCPP(int vertices)
357.    {
358.        N = vertices;
359.    }
360.
361.    OpenCPP addArc(String lab, int u, int v, float cost)
362.    {
363.        if (cost < 0)
364.            throw new Error("Graph has negative costs");
365.        arcs.addElement(new Arc(lab, u, v, cost));
366.        return this;
367.    }
368.
369.    float printCPT(int startVertex)
370.    {
371.        ChinesePostmanProblem bestGraph = null, g;
372.        float bestCost = 0, cost;
373.        int i = 0;
374.        do
375.        {
376.            g = new ChinesePostmanProblem(N + 1);
377.            for (int j = 0; j < arcs.size(); j++)

```

```

378.     {
379.         Arc it = arcs.elementAt(j);
380.         g.addArc(it.lab, it.u, it.v, it.cost);
381.     }
382.     cost = g.basicCost;
383.     g.findUnbalanced(); // initialise g.neg on original graph
384.     g.addArc("virtual start", N, startVertex, cost);
385.     g.addArc("virtual end",
386.             // graph is Eulerian if neg.length=0
387.             g.neg.length == 0 ? startVertex : g.neg[i], N, cost);
388.     g.solve();
389.     if (bestGraph == null || bestCost > g.cost())
390.     {
391.         bestCost = g.cost();
392.         bestGraph = g;
393.     }
394. }
395. while (++i < g.neg.length);
396. System.out.println("Open CPT from " + startVertex
397.                     + " (ignore virtual arcs)");
398. bestGraph.printCPT(N);
399. return cost + bestGraph.phi();
400. }
401.
402. static void test()
403. {
404.     OpenCPP G = new OpenCPP(4); // create a graph of four vertices
405.     // add the arcs for the example graph
406.     G.addArc("a", 0, 1, 1).addArc("b", 0, 2, 1).addArc("c", 1, 2, 1)
407.         .addArc("d", 1, 3, 1).addArc("e", 2, 3, 1).addArc("f", 3, 0, 1);
408.     int besti = 0;
409.     float bestCost = 0;
410.     for (int i = 0; i < 4; i++)
411.     {
412.         System.out.println("Solve from " + i);
413.         float c = G.printCPT(i);
414.         System.out.println("Cost = " + c);
415.         if (i == 0 || c < bestCost)
416.         {
417.             bestCost = c;
418.             besti = i;
419.         }
420.     }
421.     G.printCPT(besti);
422.     System.out.println("Cost = " + bestCost);
423. }
424. }

```

Output:

advertisement

```
$ javac ChinesePostmanProblem.java
$ java ChinesePostmanProblem
```

```
Take arc b from 0 to 2
Take arc e from 2 to 3
Take arc f from 3 to 0
Take arc a from 0 to 1
Take arc c from 1 to 2
Take arc e from 2 to 3
Take arc f from 3 to 0
Take arc a from 0 to 1
Take arc d from 1 to 3
Take arc f from 3 to 0
Cost = 10.0
Solve from 0
```

Open CPT from 0 (ignore virtual arcs)

Take arc 'virtual start' from 4 to 0

Take arc a from 0 to 1

Take arc d from 1 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc c from 1 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc b from 0 to 2

Take arc 'virtual end' from 2 to 4

Cost = 8.0

Solve from 1

Open CPT from 1 (ignore virtual arcs)

Take arc 'virtual start' from 4 to 1

Take arc d from 1 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc c from 1 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc b from 0 to 2

Take arc 'virtual end' from 2 to 4

Cost = 7.0

Solve from 2

Open CPT from 2 (ignore virtual arcs)

Take arc 'virtual start' from 4 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc d from 1 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc c from 1 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc b from 0 to 2

Take arc 'virtual end' from 2 to 4

Cost = 10.0

Solve from 3

Open CPT from 3 (ignore virtual arcs)

Take arc 'virtual start' from 4 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc d from 1 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc c from 1 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc b from 0 to 2

Take arc 'virtual end' from 2 to 4

Cost = 9.0

Open CPT from 1 (ignore virtual arcs)

Take arc 'virtual start' from 4 to 1

Take arc d from 1 to 3

Take arc f from 3 to 0

Take arc a from 0 to 1

Take arc c from 1 to 2

Take arc e from 2 to 3

Take arc f from 3 to 0

Take arc b from 0 to 2

Take arc 'virtual end' from 2 to 4

Cost = 7.0

314. Java Program to Check Whether an Undirected Graph Contains a Eulerian Path

[« Prev](#)

[Next »](#)

This is a java program to check whether graph contains Eulerian Cycle. The criteran Euler suggested,

1. If graph has no odd degree vertex, there is at least one Eulerian Circuit.
2. If graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.
3. If graph has more than two vertices with odd degree, there is no Eulerian Circuit or Eulerian Path.

Here is the source code of the Java Program to Check Whether an Undirected Graph Contains a Eulerian Path. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6.
7. public class UndirectedEulerPath
8. {
9.     private int[][] adjacencyMatrix;
10.    private int    numberOfNodes;
11.
12.    public UndirectedEulerPath(int    numberOfNodes, int[][] adjacencyMatrix)
13.    {
14.        this.numberOfNodes = numberOfNodes;
15.        this.adjacencyMatrix = new int[numberOfNodes + 1][numberOfNodes + 1];
16.        for (int sourceVertex = 1; sourceVertex <= numberOfNodes; sourceVertex++)
17.        {
18.            for (int destinationVertex = 1; destinationVertex <= numberOfNodes; destinationVertex++)
19.            {
20.                this.adjacencyMatrix[sourceVertex][destinationVertex] =
adjacencyMatrix[sourceVertex][destinationVertex];
21.            }
22.        }
23.    }
24.
25.    public int degree(int vertex)
26.    {
27.        int degree = 0;
28.        for (int destinationvertex = 1; destinationvertex <= numberOfNodes; destinationvertex++)
29.        {
30.            if (adjacencyMatrix[vertex][destinationvertex] == 1
31.                || adjacencyMatrix[destinationvertex][vertex] == 1)
32.            {
33.                degree++;
34.            }
35.        }
36.        return degree;
37.    }
38.
39.    public int countOddDegreeVertex()
40.    {
41.        int count = 0;
42.        for (int node = 1; node <= numberOfNodes; node++)
43.        {
44.            if (((degree(node) % 2) != 0)
45.            {
46.                count++;
47.            }
48.        }

```

```

49.     return count;
50. }
51.
52. public static void main(String... arg)
53. {
54.     int number_of_nodes;
55.     Scanner scanner = null;
56.     try
57.     {
58.         System.out.println("Enter the number of nodes in the graph");
59.         scanner = new Scanner(System.in);
60.         number_of_nodes = scanner.nextInt();
61.         int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
62.         System.out.println("Enter the adjacency matrix");
63.         for (int i = 1; i <= number_of_nodes; i++)
64.         {
65.             for (int j = 1; j <= number_of_nodes; j++)
66.             {
67.                 adjacency_matrix[i][j] = scanner.nextInt();
68.             }
69.         }
70.         //make the graph undirected
71.         for (int i = 1; i <= number_of_nodes; i++)
72.         {
73.             for (int j = 1; j <= number_of_nodes; j++)
74.             {
75.                 if (adjacency_matrix[i][j] == 1
76.                     && adjacency_matrix[j][i] == 0)
77.                 {
78.                     adjacency_matrix[j][i] = 1;
79.                 }
80.             }
81.         }
82.         DirectedEulerianCircuit path = new DirectedEulerianCircuit(number_of_nodes,
83.             adjacency_matrix);
84.         int count = path.countOddDegreeVertex();
85.         if (count == 0)
86.         {
87.             System.out
88.                 .println("As the graph has no odd degree vertex, there is at least one Eulerian Circuit.");
89.         }
90.         else if (count == 2)
91.         {
92.             System.out
93.                 .println("As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one
94. Eulerian Path.");
95.         }
96.         else
97.         {
98.             System.out
99.                 .println("As the graph has more than two vertices with odd degree, there is no Eulerian Circuit or
100. Eulerian Path.");
101.     }
102.     catch (InputMismatchException inputMismatch)
103.     {
104.         System.out.println("Wrong Input format");
105.     }
106.     scanner.close();
107. }

```

Output:

advertisement

\$ javac UndirectedEulerPath.java

```
$ java UndirectedEulerPath
```

```
Enter the number of nodes in the graph
```

```
4
```

```
Enter the adjacency matrix
```

```
0 1 1 1
```

```
1 0 1 0
```

```
1 1 0 1
```

```
1 0 1 0
```

```
As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.
```

315. Java Program to Check Whether a Directed Graph Contains a Eulerian Path

[« Prev](#)

[Next »](#)

This is a java program to check whether graph contains Eulerian Cycle. The criteran Euler suggested,

1. If graph has no odd degree vertex, there is at least one Eulerian Circuit.
2. If graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.
3. If graph has more than two vertices with odd degree, there is no Eulerian Circuit or Eulerian Path.

Here is the source code of the Java Program to Check Whether an Directed Graph Contains a Eulerian Path. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6.
7. public class DirectedEulerPath
8. {
9.     private int[][] adjacencyMatrix;
10.    private int    numberOfNodes;
11.
12.    public DirectedEulerPath(int numberOfNodes, int[][] adjacencyMatrix)
13.    {
14.        this.numberOfNodes = numberOfNodes;
15.        this.adjacencyMatrix = new int[numberOfNodes + 1][numberOfNodes + 1];
16.        for (int sourceVertex = 1; sourceVertex <= numberOfNodes; sourceVertex++)
17.        {
18.            for (int destinationVertex = 1; destinationVertex <= numberOfNodes; destinationVertex++)
19.            {
20.                this.adjacencyMatrix[sourceVertex][destinationVertex] =
adjacencyMatrix[sourceVertex][destinationVertex];
21.            }
22.        }
23.    }
24.
25.    public int degree(int vertex)
26.    {
27.        int degree = 0;
28.        for (int destinationvertex = 1; destinationvertex <= numberOfNodes; destinationvertex++)
29.        {
30.            if (adjacencyMatrix[vertex][destinationvertex] == 1
31.                || adjacencyMatrix[destinationvertex][vertex] == 1)
32.            {
33.                degree++;
34.            }
35.        }
36.        return degree;
37.    }
38.
39.    public int countOddDegreeVertex()
40.    {
41.        int count = 0;
42.        for (int node = 1; node <= numberOfNodes; node++)
43.        {
44.            if (((degree(node) % 2) != 0)
45.            {
46.                count++;
47.            }
48.        }

```

```

49.     return count;
50. }
51.
52. public static void main(String... arg)
53. {
54.     int number_of_nodes;
55.     Scanner scanner = null;
56.     try
57.     {
58.         System.out.println("Enter the number of nodes in the graph");
59.         scanner = new Scanner(System.in);
60.         number_of_nodes = scanner.nextInt();
61.         int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
62.         System.out.println("Enter the adjacency matrix");
63.         for (int i = 1; i <= number_of_nodes; i++)
64.         {
65.             for (int j = 1; j <= number_of_nodes; j++)
66.             {
67.                 adjacency_matrix[i][j] = scanner.nextInt();
68.             }
69.         }
70.         DirectedEulerPath path = new DirectedEulerPath(number_of_nodes,
71.             adjacency_matrix);
72.         int count = path.countOddDegreeVertex();
73.         if (count == 0)
74.         {
75.             System.out
76.                 .println("As the graph has no odd degree vertex, there is at least one Eulerian Circuit.");
77.         }
78.         else if (count == 2)
79.         {
80.             System.out
81.                 .println("As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one
82. Eulerian Path.");
83.         }
84.         else
85.         {
86.             System.out
87.                 .println("As the graph has more than two vertices with odd degree, there is no Eulerian Circuit or
88. Eulerian Path.");
89.         }
90.     }
91.     catch (InputMismatchException inputMismatch)
92.     {
93.         System.out.println("Wrong Input format");
94.     }
95.     scanner.close();
}

```

Output:

advertisement

```
$ javac DirectedEulerPath.java
$ java DirectedEulerPath
```

Enter the number of nodes in the graph

4

Enter the adjacency matrix

```
0 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0
```

As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.

316. Java Program to Check Whether a Directed Graph Contains a Eulerian Cycle

[« Prev](#)

[Next »](#)

This is a java program to check whether graph contains Eulerian Cycle. The criteran Euler suggested,

1. If graph has no odd degree vertex, there is at least one Eulerian Circuit.
2. If graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.
3. If graph has more than two vertices with odd degree, there is no Eulerian Circuit or Eulerian Path.

Here is the source code of the Java Program to Check Whether a Directed Graph Contains a Eulerian Cycle. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6.
7. public class DirectedEulerianCircuit
8. {
9.     private int[][] adjacencyMatrix;
10.    private int    numberOfNodes;
11.
12.    public DirectedEulerianCircuit(int numberOfNodes, int[][] adjacencyMatrix)
13.    {
14.        this.numberOfNodes = numberOfNodes;
15.        this.adjacencyMatrix = new int[numberOfNodes + 1][numberOfNodes + 1];
16.        for (int sourceVertex = 1; sourceVertex <= numberOfNodes; sourceVertex++)
17.        {
18.            for (int destinationVertex = 1; destinationVertex <= numberOfNodes; destinationVertex++)
19.            {
20.                this.adjacencyMatrix[sourceVertex][destinationVertex] =
adjacencyMatrix[sourceVertex][destinationVertex];
21.            }
22.        }
23.    }
24.
25.    public int degree(int vertex)
26.    {
27.        int degree = 0;
28.        for (int destinationvertex = 1; destinationvertex <= numberOfNodes; destinationvertex++)
29.        {
30.            if (adjacencyMatrix[vertex][destinationvertex] == 1
31.                || adjacencyMatrix[destinationvertex][vertex] == 1)
32.            {
33.                degree++;
34.            }
35.        }
36.        return degree;
37.    }
38.
39.    public int countOddDegreeVertex()
40.    {
41.        int count = 0;
42.        for (int node = 1; node <= numberOfNodes; node++)
43.        {
44.            if (((degree(node) % 2) != 0)
45.            {
46.                count++;
47.            }
48.        }

```

```

49.     return count;
50. }
51.
52. public static void main(String... arg)
53. {
54.     int number_of_nodes;
55.     Scanner scanner = null;
56.     try
57.     {
58.         System.out.println("Enter the number of nodes in the graph");
59.         scanner = new Scanner(System.in);
60.         number_of_nodes = scanner.nextInt();
61.         int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
62.         System.out.println("Enter the adjacency matrix");
63.         for (int i = 1; i <= number_of_nodes; i++)
64.         {
65.             for (int j = 1; j <= number_of_nodes; j++)
66.             {
67.                 adjacency_matrix[i][j] = scanner.nextInt();
68.             }
69.         }
70.         //make the graph undirected
71.         for (int i = 1; i <= number_of_nodes; i++)
72.         {
73.             for (int j = 1; j <= number_of_nodes; j++)
74.             {
75.                 if (adjacency_matrix[i][j] == 1
76.                     && adjacency_matrix[j][i] == 0)
77.                 {
78.                     adjacency_matrix[j][i] = 1;
79.                 }
80.             }
81.         }
82.         UndirectedEulerPath path = new UndirectedEulerPath(number_of_nodes,
83.             adjacency_matrix);
84.         int count = path.countOddDegreeVertex();
85.         if (count == 0)
86.         {
87.             System.out
88.                 .println("As the graph has no odd degree vertex, there is at least one Eulerian Circuit.");
89.         }
90.         else if (count == 2)
91.         {
92.             System.out
93.                 .println("As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one
94. Eulerian Path.");
95.         }
96.         else
97.         {
98.             System.out
99.                 .println("As the graph has more than two vertices with odd degree, there is no Eulerian Circuit or
100. Eulerian Path.");
101.     }
102.     catch (InputMismatchException inputMismatch)
103.     {
104.         System.out.println("Wrong Input format");
105.     }
106. }
107. }
```

Output:

advertisement

\$ javac DirectedEulerianCircuit.java

```
$ java DirectedEulerianCircuit
```

```
Enter the number of nodes in the graph
```

```
6
```

```
Enter the adjacency matrix
```

```
0 1 0 0 0 0  
0 0 1 1 0 0  
0 0 0 0 0 0  
0 0 0 0 1 0  
0 0 0 0 0 1  
0 0 1 0 0 0
```

```
As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path
```

317. Java Program to Check Whether an Undirected Graph Contains a Eulerian Cycle

[« Prev](#)

[Next »](#)

This is a java program to check whether graph contains Eulerian Cycle. The criteran Euler suggested,

1. If graph has no odd degree vertex, there is at least one Eulerian Circuit.
2. If graph has two vertices with odd degree, there is no Eulerian Circuit but at least one Eulerian Path.
3. If graph has more than two vertices with odd degree, there is no Eulerian Circuit or Eulerian Path.

Here is the source code of the Java Program to Check Whether an Undirected Graph Contains a Eulerian Cycle. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6.
7. public class UndirectedEulerianCircuit
8. {
9.     private int[][] adjacencyMatrix;
10.    private int    numberOfNodes;
11.
12.    public UndirectedEulerianCircuit(int    numberOfNodes, int[][] adjacencyMatrix)
13.    {
14.        this.numberOfNodes = numberOfNodes;
15.        this.adjacencyMatrix = new int[numberOfNodes + 1][numberOfNodes + 1];
16.        for (int sourceVertex = 1; sourceVertex <= numberOfNodes; sourceVertex++)
17.        {
18.            for (int destinationVertex = 1; destinationVertex <= numberOfNodes; destinationVertex++)
19.            {
20.                this.adjacencyMatrix[sourceVertex][destinationVertex] =
adjacencyMatrix[sourceVertex][destinationVertex];
21.            }
22.        }
23.    }
24.
25.    public int degree(int vertex)
26.    {
27.        int degree = 0;
28.        for (int destinationvertex = 1; destinationvertex <= numberOfNodes; destinationvertex++)
29.        {
30.            if (adjacencyMatrix[vertex][destinationvertex] == 1
31.                || adjacencyMatrix[destinationvertex][vertex] == 1)
32.            {
33.                degree++;
34.            }
35.        }
36.        return degree;
37.    }
38.
39.    public int countOddDegreeVertex()
40.    {
41.        int count = 0;
42.        for (int node = 1; node <= numberOfNodes; node++)
43.        {
44.            if ((degree(node) % 2) != 0)
45.            {
46.                count++;
47.            }
48.        }
49.    }
50.
51.    public void printGraph()
52.    {
53.        for (int i = 1; i <= numberOfNodes; i++)
54.        {
55.            System.out.print(i + " : ");
56.            for (int j = 1; j <= numberOfNodes; j++)
57.            {
58.                if (adjacencyMatrix[i][j] == 1)
59.                    System.out.print(j + " ");
60.            }
61.            System.out.println();
62.        }
63.    }
64.
65.    public static void main(String[] args)
66.    {
67.        Scanner scanner = new Scanner(System.in);
68.        UndirectedEulerianCircuit circuit = new UndirectedEulerianCircuit();
69.        System.out.print("Enter number of nodes: ");
70.        int    n = scanner.nextInt();
71.        System.out.print("Enter adjacency matrix: ");
72.        int[][] matrix = new int[n + 1][n + 1];
73.        for (int i = 1; i <= n; i++)
74.        {
75.            for (int j = 1; j <= n; j++)
76.            {
77.                matrix[i][j] = scanner.nextInt();
78.            }
79.        }
80.        circuit.adjacencyMatrix = matrix;
81.        System.out.println("Graph is: ");
82.        circuit.printGraph();
83.        System.out.println("Number of odd degree vertices: " +
circuit.countOddDegreeVertex());
84.        if (circuit.countOddDegreeVertex() > 2)
85.            System.out.println("Graph does not contain Eulerian Circuit");
86.        else if (circuit.countOddDegreeVertex() == 2)
87.            System.out.println("Graph contains Eulerian Path");
88.        else
89.            System.out.println("Graph contains Eulerian Circuit");
90.    }
91.
92. }
```

```

48.     }
49.     return count;
50. }
51.
52. public static void main(String... arg)
53. {
54.     int number_of_nodes;
55.     Scanner scanner = null;
56.     try
57.     {
58.         System.out.println("Enter the number of nodes in the graph");
59.         scanner = new Scanner(System.in);
60.         number_of_nodes = scanner.nextInt();
61.         int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
62.         System.out.println("Enter the adjacency matrix");
63.         for (int i = 1; i <= number_of_nodes; i++)
64.         {
65.             for (int j = 1; j <= number_of_nodes; j++)
66.             {
67.                 adjacency_matrix[i][j] = scanner.nextInt();
68.             }
69.         }
70.         //make the graph undirected
71.         for (int i = 1; i <= number_of_nodes; i++)
72.         {
73.             for (int j = 1; j <= number_of_nodes; j++)
74.             {
75.                 if (adjacency_matrix[i][j] == 1
76.                     && adjacency_matrix[j][i] == 0)
77.                 {
78.                     adjacency_matrix[j][i] = 1;
79.                 }
80.             }
81.         }
82.         UndirectedEulerPath path = new UndirectedEulerPath(number_of_nodes,
83.             adjacency_matrix);
84.         int count = path.countOddDegreeVertex();
85.         if (count == 0)
86.         {
87.             System.out
88.                 .println("As the graph has no odd degree vertex, there is at least one Eulerian Circuit.");
89.         }
90.         else if (count == 2)
91.         {
92.             System.out
93.                 .println("As the graph has two vertices with odd degree, there is no Eulerian Circuit but at least one
94. Eulerian Path.");
95.         }
96.         else
97.         {
98.             System.out
99.                 .println("As the graph has more than two vertices with odd degree, there is no Eulerian Circuit or
100. Eulerian Path.");
101.        }
102.        catch (InputMismatchException inputMismatch)
103.        {
104.            System.out.println("Wrong Input format");
105.        }
106.    }
107.}

```

Output:

advertisement

```
$ javac UndirectedEulerianCircuit.java  
$ java UndirectedEulerianCircuit
```

Enter the number of nodes in the graph

6

Enter the adjacency matrix

```
0 1 0 0 0 0  
1 0 1 1 0 0  
0 1 0 0 0 1  
0 1 0 0 1 1  
0 0 0 1 0 1  
0 0 1 1 1 0
```

As the graph has more than two vertices with odd degree, there is no Eulerian Circuit or Eulerian Path.

318. Java Program to Implement the Edmond's Algorithm for Maximum Cardinality Matching

[« Prev](#)

[Next »](#)

This is a java program to implement Edmond's Algorithm for maximum cardinality matching. In graph theory, a branch of mathematics, Edmonds' algorithm or Chu–Liu/Edmonds' algorithm is an algorithm for finding a maximum or minimum optimum branchings. This is similar to the minimum spanning tree problem which concerns undirected graphs. However, when nodes are connected by weighted edges that are directed, a minimum spanning tree algorithm cannot be used.

Here is the source code of the Java Program to Implement the Edmond's Algorithm for Maximum Cardinality Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.ArrayList;
5. import java.util.Arrays;
6. import java.util.List;
7. import java.util.Scanner;
8.
9. public class EdmondsMaximumCardinalityMatching
10. {
11.     static int lca(int[] match, int[] base, int[] p, int a, int b)
12.     {
13.         boolean[] used = new boolean[match.length];
14.         while (true)
15.         {
16.             a = base[a];
17.             used[a] = true;
18.             if (match[a] == -1)
19.                 break;
20.             a = p[match[a]];
21.         }
22.         while (true)
23.         {
24.             b = base[b];
25.             if (used[b])
26.                 return b;
27.             b = p[match[b]];
28.         }
29.     }
30.
31.     static void markPath(int[] match, int[] base, boolean[] blossom, int[] p,
32.             int v, int b, int children)
33.     {
34.         for (; base[v] != b; v = p[match[v]])
35.         {
36.             blossom[base[v]] = blossom[base[match[v]]] = true;
37.             p[v] = children;
38.             children = match[v];
39.         }
40.     }
41.
42.     static int findPath(List<Integer>[] graph, int[] match, int[] p, int root)
43.     {
44.         int n = graph.length;
45.         boolean[] used = new boolean[n];
46.         Arrays.fill(p, -1);
47.         int[] base = new int[n];
```

```

48.     for (int i = 0; i < n; ++i)
49.         base[i] = i;
50.     used[root] = true;
51.     int qh = 0;
52.     int qt = 0;
53.     int[] q = new int[n];
54.     q[qt++] = root;
55.     while (qh < qt)
56.     {
57.         int v = q[qh++];
58.         for (int to : graph[v])
59.         {
60.             if (base[v] == base[to] || match[v] == to)
61.                 continue;
62.             if (to == root || match[to] != -1 && p[match[to]] != -1)
63.             {
64.                 int curbase = lca(match, base, p, v, to);
65.                 boolean[] blossom = new boolean[n];
66.                 markPath(match, base, blossom, p, v, curbase, to);
67.                 markPath(match, base, blossom, p, to, curbase, v);
68.                 for (int i = 0; i < n; ++i)
69.                     if (blossom[base[i]])
70.                     {
71.                         base[i] = curbase;
72.                         if (!used[i])
73.                         {
74.                             used[i] = true;
75.                             q[qt++] = i;
76.                         }
77.                     }
78.             }
79.             else if (p[to] == -1)
80.             {
81.                 p[to] = v;
82.                 if (match[to] == -1)
83.                     return to;
84.                 to = match[to];
85.                 used[to] = true;
86.                 q[qt++] = to;
87.             }
88.         }
89.     }
90.     return -1;
91. }
92.
93. public static int maxMatching(List<Integer>[] graph)
94. {
95.     int n = graph.length;
96.     int[] match = new int[n];
97.     Arrays.fill(match, -1);
98.     int[] p = new int[n];
99.     for (int i = 0; i < n; ++i)
100.    {
101.        if (match[i] == -1)
102.        {
103.            int v = findPath(graph, match, p, i);
104.            while (v != -1)
105.            {
106.                int pv = p[v];
107.                int ppv = match[pv];
108.                match[v] = pv;
109.                match[pv] = v;
110.                v = ppv;
111.            }
112.        }
113.    }

```

```

114.     int matches = 0;
115.     for (int i = 0; i < n; ++i)
116.         if (match[i] != -1)
117.             ++matches;
118.     return matches / 2;
119. }
120.
121. @SuppressWarnings("unchecked")
122. public static void main(String[] args)
123. {
124.     Scanner sc = new Scanner(System.in);
125.     System.out.println("Enter the number of vertices: ");
126.     int v = sc.nextInt();
127.     System.out.println("Enter the number of edges: ");
128.     int e = sc.nextInt();
129.     List<Integer>[] g = new List[v];
130.     for (int i = 0; i < v; i++)
131.     {
132.         g[i] = new ArrayList<Integer>();
133.     }
134.     System.out.println("Enter all the edges: <from> <to>");
135.     for (int i = 0; i < e; i++)
136.     {
137.         g[sc.nextInt()].add(sc.nextInt());
138.     }
139.     System.out.println("Maximum matching for the given graph is: "
140.                         + maxMatching(g));
141.     sc.close();
142. }
143.

```

Output:

advertisement

```
$ javac EdmondsMaximumCardinalityMatching.java
$ java EdmondsMaximumCardinalityMatching
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter all the edges: <from> <to>

0 1

1 2

1 3

3 4

4 5

5 3

5 2

Maximum matching for the given graph is: 3

319. Java Program to Implement the Hungarian Algorithm for Bipartite Matching

[« Prev](#)

[Next »](#)

This is a java program to implement Hungarian Algorithm for Bipartite Matching. The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal-dual methods.

Here is the source code of the Java Program to Implement the Hungarian Algorithm for Bipartite Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.  
2. package com.hinguapps.graph;  
3.  
4. import java.util.Arrays;  
5. import java.util.Scanner;  
6.  
7. public class HungarianBipartiteMatching  
8. {  
9.     private final double[][] costMatrix;  
10.    private final int    rows, cols, dim;  
11.    private final double[] labelByWorker, labelByJob;  
12.    private final int[]   minSlackWorkerByJob;  
13.    private final double[] minSlackValueByJob;  
14.    private final int[]   matchJobByWorker, matchWorkerByJob;  
15.    private final int[]   parentWorkerByCommittedJob;  
16.    private final boolean[] committedWorkers;  
17.  
18.    public HungarianBipartiteMatching(double[][] costMatrix)  
19.    {  
20.        this.dim = Math.max(costMatrix.length, costMatrix[0].length);  
21.        this.rows = costMatrix.length;  
22.        this.cols = costMatrix[0].length;  
23.        this.costMatrix = new double[this.dim][this.dim];  
24.        for (int w = 0; w < this.dim; w++)  
25.        {  
26.            if (w < costMatrix.length)  
27.            {  
28.                if (costMatrix[w].length != this.cols)  
29.                {  
30.                    throw new IllegalArgumentException("Irregular cost matrix");  
31.                }  
32.                this.costMatrix[w] = Arrays.copyOf(costMatrix[w], this.dim);  
33.            }  
34.            else  
35.            {  
36.                this.costMatrix[w] = new double[this.dim];  
37.            }  
38.        }  
39.        labelByWorker = new double[this.dim];  
40.        labelByJob = new double[this.dim];  
41.        minSlackWorkerByJob = new int[this.dim];  
42.        minSlackValueByJob = new double[this.dim];  
43.        committedWorkers = new boolean[this.dim];  
44.        parentWorkerByCommittedJob = new int[this.dim];  
45.        matchJobByWorker = new int[this.dim];  
46.        Arrays.fill(matchJobByWorker, -1);  
47.        matchWorkerByJob = new int[this.dim];  
48.        Arrays.fill(matchWorkerByJob, -1);  
49.    }  
50.
```

```

51. protected void computeInitialFeasibleSolution()
52. {
53.     for (int j = 0; j < dim; j++)
54.     {
55.         labelByJob[j] = Double.POSITIVE_INFINITY;
56.     }
57.     for (int w = 0; w < dim; w++)
58.     {
59.         for (int j = 0; j < dim; j++)
60.         {
61.             if (costMatrix[w][j] < labelByJob[j])
62.             {
63.                 labelByJob[j] = costMatrix[w][j];
64.             }
65.         }
66.     }
67. }
68.
69. public int[] execute()
70. {
71.     /*
72.      * Heuristics to improve performance: Reduce rows and columns by their
73.      * smallest element, compute an initial non-zero dual feasible solution
74.      * and
75.      * create a greedy matching from workers to jobs of the cost matrix.
76.      */
77.     reduce();
78.     computeInitialFeasibleSolution();
79.     greedyMatch();
80.     int w = fetchUnmatchedWorker();
81.     while (w < dim)
82.     {
83.         initializePhase(w);
84.         executePhase();
85.         w = fetchUnmatchedWorker();
86.     }
87.     int[] result = Arrays.copyOf(matchJobByWorker, rows);
88.     for (w = 0; w < result.length; w)
89.     {
90.         if (result[w] >= cols)
91.         {
92.             result[w] = -1;
93.         }
94.     }
95.     return result;
96. }
97.
98. protected void executePhase()
99. {
100.    while (true)
101.    {
102.        int minSlackWorker = -1, minSlackJob = -1;
103.        double minSlackValue = Double.POSITIVE_INFINITY;
104.        for (int j = 0; j < dim; j++)
105.        {
106.            if (parentWorkerByCommittedJob[j] == -1)
107.            {
108.                if (minSlackValueByJob[j] < minSlackValue)
109.                {
110.                    minSlackValue = minSlackValueByJob[j];
111.                    minSlackWorker = minSlackWorkerByJob[j];
112.                    minSlackJob = j;
113.                }
114.            }
115.        }
116.        if (minSlackValue > 0)

```

```

117.     {
118.         updateLabeling(minSlackValue);
119.     }
120.     parentWorkerByCommittedJob[minSlackJob] = minSlackWorker;
121.     if (matchWorkerByJob[minSlackJob] == -1)
122.     {
123.         /*
124.          * An augmenting path has been found.
125.         */
126.         int committedJob = minSlackJob;
127.         int parentWorker = parentWorkerByCommittedJob[committedJob];
128.         while (true)
129.         {
130.             int temp = matchJobByWorker[parentWorker];
131.             match(parentWorker, committedJob);
132.             committedJob = temp;
133.             if (committedJob == -1)
134.             {
135.                 break;
136.             }
137.             parentWorker = parentWorkerByCommittedJob[committedJob];
138.         }
139.         return;
140.     }
141.     else
142.     {
143.         /*
144.          * Update slack values since we increased the size of the
145.          * committed
146.          * workers set.
147.         */
148.         int worker = matchWorkerByJob[minSlackJob];
149.         committedWorkers[worker] = true;
150.         for (int j = 0; j < dim; j++)
151.         {
152.             if (parentWorkerByCommittedJob[j] == -1)
153.             {
154.                 double slack = costMatrix[worker][j]
155.                     - labelByWorker[worker] - labelByJob[j];
156.                 if (minSlackValueByJob[j] > slack)
157.                 {
158.                     minSlackValueByJob[j] = slack;
159.                     minSlackWorkerByJob[j] = worker;
160.                 }
161.             }
162.         }
163.     }
164. }
165. }
166.
167. protected int fetchUnmatchedWorker()
168. {
169.     int w;
170.     for (w = 0; w < dim; w++)
171.     {
172.         if (matchJobByWorker[w] == -1)
173.         {
174.             break;
175.         }
176.     }
177.     return w;
178. }
179.
180. protected void greedyMatch()
181. {
182.     for (int w = 0; w < dim; w++)

```

```

183.     {
184.         for (int j = 0; j < dim; j++)
185.         {
186.             if (matchJobByWorker[w] == -1
187.                 && matchWorkerByJob[j] == -1
188.                 && costMatrix[w][j] - labelByWorker[w] - labelByJob[j] == 0)
189.             {
190.                 match(w, j);
191.             }
192.         }
193.     }
194. }
195.
196. protected void initializePhase(int w)
197. {
198.     Arrays.fill(committedWorkers, false);
199.     Arrays.fill(parentWorkerByCommittedJob, -1);
200.     committedWorkers[w] = true;
201.     for (int j = 0; j < dim; j++)
202.     {
203.         minSlackValueByJob[j] = costMatrix[w][j] - labelByWorker[w]
204.             - labelByJob[j];
205.         minSlackWorkerByJob[j] = w;
206.     }
207. }
208.
209. protected void match(int w, int j)
210. {
211.     matchJobByWorker[w] = j;
212.     matchWorkerByJob[j] = w;
213. }
214.
215. protected void reduce()
216. {
217.     for (int w = 0; w < dim; w++)
218.     {
219.         double min = Double.POSITIVE_INFINITY;
220.         for (int j = 0; j < dim; j++)
221.         {
222.             if (costMatrix[w][j] < min)
223.             {
224.                 min = costMatrix[w][j];
225.             }
226.         }
227.         for (int j = 0; j < dim; j++)
228.         {
229.             costMatrix[w][j] -= min;
230.         }
231.     }
232.     double[] min = new double[dim];
233.     for (int j = 0; j < dim; j++)
234.     {
235.         min[j] = Double.POSITIVE_INFINITY;
236.     }
237.     for (int w = 0; w < dim; w++)
238.     {
239.         for (int j = 0; j < dim; j++)
240.         {
241.             if (costMatrix[w][j] < min[j])
242.             {
243.                 min[j] = costMatrix[w][j];
244.             }
245.         }
246.     }
247.     for (int w = 0; w < dim; w++)
248.     {

```

```

249.         for (int j = 0; j < dim; j++)
250.         {
251.             costMatrix[w][j] -= min[j];
252.         }
253.     }
254. }
255.
256. protected void updateLabeling(double slack)
257. {
258.     for (int w = 0; w < dim; w++)
259.     {
260.         if (committedWorkers[w])
261.         {
262.             labelByWorker[w] += slack;
263.         }
264.     }
265.     for (int j = 0; j < dim; j++)
266.     {
267.         if (parentWorkerByCommittedJob[j] != -1)
268.         {
269.             labelByJob[j] -= slack;
270.         }
271.         else
272.         {
273.             minSlackValueByJob[j] -= slack;
274.         }
275.     }
276. }
277.
278. public static void main(String[] args)
279. {
280.     Scanner sc = new Scanner(System.in);
281.     System.out.println("Enter the dimensions of the cost matrix: ");
282.     System.out.println("r:");
283.     int r = sc.nextInt();
284.     System.out.println("c:");
285.     int c = sc.nextInt();
286.     System.out.println("Enter the cost matrix: <row wise>");
287.     double[][] cost = new double[r][c];
288.     for (int i = 0; i < r; i++)
289.     {
290.         for (int j = 0; j < c; j++)
291.         {
292.             cost[i][j] = sc.nextDouble();
293.         }
294.     }
295.     HungarianBipartiteMatching hbm = new HungarianBipartiteMatching(cost);
296.     int[] result = hbm.execute();
297.     System.out.println("Bipartite Matching: " + Arrays.toString(result));
298.     sc.close();
299. }
300. }
```

Output:

```
$ javac HungarianBipartiteMatching.java
$ java HungarianBipartiteMatching
```

Enter the dimensions of the cost matrix:

r: 4

c: 4

Enter the cost matrix: <row wise>

82	83	69	92
77	37	49	92
11	69	5	86
8	9	98	23

Bipartite Matching: [2, 1, 0, 3] //worker 1 should perform job 3, worker 2 should perform job 2 and so on..

320. Java Program to Solve a Matching Problem for a Given Specific Case

[« Prev](#)

[Next »](#)

This is a java program to solve a matching problem for stable marriage matching problem. In mathematics, economics, and computer science, the stable marriage problem (SMP) is the problem of finding a stable matching between two sets of elements given a set of preferences for each element. A matching is a mapping from the elements of one set to the elements of the other set. A matching is stable whenever it is not the case that both:

1. some given element A of the first matched set prefers some given element B of the second matched set over the element to which A is already matched, and
2. B also prefers A over the element to which B is already matched

In other words, a matching is stable when there does not exist any alternative pairing (A, B) in which both A and B are individually better off than they would be with the element to which they are currently matched.

advertisement

The stable marriage problem is commonly stated as: Given n men and n women, where each person has ranked all members of the opposite gender with a unique number between 1 and n in order of preference, marry the men and women together such that there are no two people of opposite gender who would both rather have each other than their current partners. If there are no such people, all the marriages are “stable”.

Here is the source code of the Java Program to Solve a Matching Problem for a Given Specific Case. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.*;
5.
6. public class MatchingProblemStableMarriage
7. {
8.     static List<String> guys = Arrays.asList(new String[] { "abe", "bob",
9.         "col", "dan", "ed", "fred", "gav", "hal", "ian", "jon" });
10.    static List<String> girls = Arrays.asList(new String[] { "abi", "bea",
11.        "cath", "dee", "eve", "fay", "gay", "hope", "ivy", "jan" });
12.    static Map<String, List<String>> guyPrefers = new HashMap<String, List<String>>()
13.    {
14.        private static final long serialVersionUID = 1L;
15.        {
16.            put("abe", Arrays.asList("abi", "eve", "cath", "ivy", "jan", "dee",
17.                "fay", "bea", "hope", "gay"));
18.            put("bob", Arrays.asList("cath", "hope", "abi", "dee", "eve",
19.                "fay", "bea", "jan", "ivy", "gay"));
20.            put("col", Arrays.asList("hope", "eve", "abi", "dee", "bea", "fay",
21.                "ivy", "gay", "cath", "jan"));
22.            put("dan", Arrays.asList("ivy", "fay", "dee", "gay", "hope", "eve",
23.                "jan", "bea", "cath", "abi"));
24.            put("ed", Arrays.asList("jan", "dee", "bea", "cath", "fay", "eve",
25.                "abi", "ivy", "hope", "gay"));
26.            put("fred", Arrays.asList("bea", "abi", "dee", "gay", "eve", "ivy",
27.                "cath", "jan", "hope", "fay"));
28.            put("gav", Arrays.asList("gay", "eve", "ivy", "bea", "cath", "abi",
29.                "dee", "hope", "jan", "fay"));
30.            put("hal", Arrays.asList("abi", "eve", "hope", "fay", "ivy",
31.                "cath", "jan", "bea", "gay", "dee")));
32.    }
33. }
```

```

32.     put("ian", Arrays.asList("hope", "cath", "dee", "gay", "bea",
33.         "abi", "fay", "ivy", "jan", "eve")));
34.     put("jon", Arrays.asList("abi", "fay", "jan", "gay", "eve", "bea",
35.         "dee", "cath", "ivy", "hope")));
36.   }
37. };
38. static Map<String, List<String>> girlPrefers = new HashMap<String, List<String>>()
39.
40. private static final long serialVersionUID = 1L;
41.
42. {
43.     put("abi", Arrays.asList("bob", "fred", "jon", "gav", "ian", "abe",
44.         "dan", "ed", "col", "hal"));
45.     put("bea", Arrays.asList("bob", "abe", "col", "fred", "gav", "dan",
46.         "ian", "ed", "jon", "hal"));
47.     put("cath", Arrays.asList("fred", "bob", "ed", "gav", "hal", "col",
48.         "ian", "abe", "dan", "jon"));
49.     put("dee", Arrays.asList("fred", "jon", "col", "abe", "ian", "hal",
50.         "gav", "dan", "bob", "ed"));
51.     put("eve", Arrays.asList("jon", "hal", "fred", "dan", "abe", "gav",
52.         "col", "ed", "ian", "bob"));
53.     put("fay", Arrays.asList("bob", "abe", "ed", "ian", "jon", "dan",
54.         "fred", "gav", "col", "hal"));
55.     put("gay", Arrays.asList("jon", "gav", "hal", "fred", "bob", "abe",
56.         "col", "ed", "dan", "ian"));
57.     put("hope", Arrays.asList("gav", "jon", "bob", "abe", "ian", "dan",
58.         "hal", "ed", "col", "fred"));
59.     put("ivy", Arrays.asList("ian", "col", "hal", "gav", "fred", "bob",
60.         "abe", "ed", "jon", "dan"));
61.     put("jan", Arrays.asList("ed", "hal", "gav", "abe", "bob", "jon",
62.         "col", "ian", "fred", "dan));
63.   };
64.
65. public static void main(String[] args)
66. {
67.     Map<String, String> matches = match(guys, guyPrefers, girlPrefers);
68.     for (Map.Entry<String, String> couple : matches.entrySet())
69.     {
70.         System.out.println(couple.getKey() + " is engaged to "
71.             + couple.getValue());
72.     }
73.     if (checkMatches(guys, girls, matches, guyPrefers, girlPrefers))
74.     {
75.         System.out.println("Marriages are stable");
76.     }
77.     else
78.     {
79.         System.out.println("Marriages are unstable");
80.     }
81.     String tmp = matches.get(girls.get(0));
82.     matches.put(girls.get(0), matches.get(girls.get(1)));
83.     matches.put(girls.get(1), tmp);
84.     System.out.println(girls.get(0) + " and " + girls.get(1)
85.         + " have switched partners");
86.     if (checkMatches(guys, girls, matches, guyPrefers, girlPrefers))
87.     {
88.         System.out.println("Marriages are stable");
89.     }
90.     else
91.     {
92.         System.out.println("Marriages are unstable");
93.     }
94. }
95.
96. private static Map<String, String> match(List<String> guys,
97.     Map<String, List<String>> guyPrefers,

```

```

98.     Map<String, List<String>> girlPrefers
99. {
100.    Map<String, String> engagedTo = new TreeMap<String, String>();
101.    List<String> freeGuys = new LinkedList<String>();
102.    freeGuys.addAll(guys);
103.    while (!freeGuys.isEmpty())
104.    {
105.        String thisGuy = freeGuys.remove(0); // get a load of THIS guy
106.        List<String> thisGuyPrefers = guyPrefers.get(thisGuy);
107.        for (String girl : thisGuyPrefers)
108.        {
109.            if (engagedTo.get(girl) == null)
110.                // girl is free
111.                engagedTo.put(girl, thisGuy); // awww
112.                break;
113.            }
114.        else
115.        {
116.            String otherGuy = engagedTo.get(girl);
117.            List<String> thisGirlPrefers = girlPrefers.get(girl);
118.            if (thisGirlPrefers.indexOf(thisGuy) < thisGirlPrefers
119.                .indexOf(otherGuy))
120.            {
121.                // this girl prefers this guy to the guy she's engaged
122.                // to
123.                engagedTo.put(girl, thisGuy);
124.                freeGuys.add(otherGuy);
125.                break;
126.            } // else no change...keep looking for this guy
127.        }
128.    }
129. }
130. return engagedTo;
131. }
132.
133. private static boolean checkMatches(List<String> guys, List<String> girls,
134.     Map<String, String> matches, Map<String, List<String>> guyPrefers,
135.     Map<String, List<String>> girlPrefers)
136. {
137.     if (!matches.keySet().containsAll(girls))
138.     {
139.         return false;
140.     }
141.     if (!matches.values().containsAll(guys))
142.     {
143.         return false;
144.     }
145.     Map<String, String> invertedMatches = new TreeMap<String, String>();
146.     for (Map.Entry<String, String> couple : matches.entrySet())
147.     {
148.         invertedMatches.put(couple.getValue(), couple.getKey());
149.     }
150.     for (Map.Entry<String, String> couple : matches.entrySet())
151.     {
152.         List<String> shePrefers = girlPrefers.get(couple.getKey());
153.         List<String> sheLikesBetter = new LinkedList<String>();
154.         sheLikesBetter.addAll(shePrefers.subList(0,
155.             shePrefers.indexOf(couple.getValue())));
156.         List<String> hePrefers = guyPrefers.get(couple.getValue());
157.         List<String> heLikesBetter = new LinkedList<String>();
158.         heLikesBetter.addAll(hePrefers.subList(0,
159.             hePrefers.indexOf(couple.getKey())));
160.         for (String guy : sheLikesBetter)
161.         {
162.             String guysFinace = invertedMatches.get(guy);
163.             List<String> thisGuyPrefers = guyPrefers.get(guy);

```

```

164.     if (thisGuyPrefers.indexOf(guysFinace) > thisGuyPrefers
165.         .indexOf(couple.getKey()))
166.     {
167.         System.out.printf("%s likes %s better than %s and %s"
168.             + " likes %s better than their current partner\n",
169.             couple.getKey(), guy, couple.getValue(), guy,
170.             couple.getKey());
171.         return false;
172.     }
173. }
174. for (String girl : heLikesBetter)
175. {
176.     String girlsFinace = matches.get(girl);
177.     List<String> thisGirlPrefers = girlPrefers.get(girl);
178.     if (thisGirlPrefers.indexOf(girlsFinace) > thisGirlPrefers
179.         .indexOf(couple.getValue()))
180.     {
181.         System.out.printf("%s likes %s better than %s and %s"
182.             + " likes %s better than their current partner\n",
183.             couple.getValue(), girl, couple.getKey(), girl,
184.             couple.getValue());
185.         return false;
186.     }
187. }
188. }
189. return true;
190. }
191. }
```

Output:

advertisement

```
$ javac MatchingProblemStableMarriage.java
$ java MatchingProblemStableMarriage
```

```

abi is engaged to jon
bea is engaged to fred
cath is engaged to bob
dee is engaged to col
eve is engaged to hal
fay is engaged to dan
gay is engaged to gav
hope is engaged to ian
ivy is engaged to abe
jan is engaged to ed
Marriages are stable
abi and bea have switched partners
fred likes bea better than abi and bea likes fred better than their current partner
Marriages are unstable
```

321. Java Program to Find Shortest Path Between All Vertices Using Floyd-Warshall's Algorithm

[« Prev](#)

[Next »](#)

This is a java program to find shortest path between all vertices using FLoyd-Warshall's algorithm. In computer science, the Floyd–Warshall algorithm (also known as Floyd's algorithm, Roy–Warshall algorithm, Roy–Floyd algorithm, or the WFI algorithm) is a graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles, see below) and also for finding transitive closure of a relation R. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pairs of vertices, though it does not return details of the paths themselves.

Here is the source code of the Java Program to Find Shortest Path Between All Vertices Using Floyd-Warshall's Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.graph;
3. 
4. import java.util.Scanner;
5. 
6. public class FloydWarshallShortestPath
7. {
8.     private int      distancematrix[][];
9.     private int      numberofvertices;
10.    public static final int INFINITY = 999;
11. 
12.   public FloydWarshallShortestPath(int numberofvertices)
13.   {
14.       distancematrix = new int[numberofvertices + 1][numberofvertices + 1];
15.       this.numberofvertices = numberofvertices;
16.   }
17. 
18.   public void floydwarshall(int adjacencymatrix[][])
19.   {
20.       for (int source = 1; source <= numberofvertices; source++)
21.       {
22.           for (int destination = 1; destination <= numberofvertices; destination++)
23.           {
24.               distancematrix[source][destination] = adjacencymatrix[source][destination];
25.           }
26.       }
27.       for (int intermediate = 1; intermediate <= numberofvertices; intermediate++)
28.       {
29.           for (int source = 1; source <= numberofvertices; source++)
30.           {
31.               for (int destination = 1; destination <= numberofvertices; destination++)
32.               {
33.                   if (distancematrix[source][intermediate]
34.                       + distancematrix[intermediate][destination] < distancematrix[source][destination])
35.                       distancematrix[source][destination] = distancematrix[source][intermediate]
36.                                     + distancematrix[intermediate][destination];
37.               }
38.           }
39.       }
40.       for (int source = 1; source <= numberofvertices; source++)
41.           System.out.print("\t" + source);
42.       System.out.println();
43.       for (int source = 1; source <= numberofvertices; source++)
44.       {
45.           System.out.print(source + "\t");
```

```

46.     for (int destination = 1; destination <= numberofvertices; destination++)
47.     {
48.         System.out.print(distancematrix[source][destination] + "\t");
49.     }
50.     System.out.println();
51. }
52. }
53.
54. public static void main(String... arg)
55. {
56.     int adjacency_matrix[][];
57.     int numberofvertices;
58.     Scanner scan = new Scanner(System.in);
59.     System.out.println("Enter the number of vertices");
60.     numberofvertices = scan.nextInt();
61.     adjacency_matrix = new int[numberofvertices + 1][numberofvertices + 1];
62.     System.out.println("Enter the Weighted Matrix for the graph");
63.     for (int source = 1; source <= numberofvertices; source++)
64.     {
65.         for (int destination = 1; destination <= numberofvertices; destination++)
66.         {
67.             adjacency_matrix[source][destination] = scan.nextInt();
68.             if (source == destination)
69.             {
70.                 adjacency_matrix[source][destination] = 0;
71.                 continue;
72.             }
73.             if (adjacency_matrix[source][destination] == 0)
74.             {
75.                 adjacency_matrix[source][destination] = INFINITY;
76.             }
77.         }
78.     }
79.     System.out.println("The Transitive Closure of the Graph");
80.     FloydWarshallShortestPath floydwarshall = new FloydWarshallShortestPath(
81.         numberofvertices);
82.     floydwarshall.floydwarshall(adjacency_matrix);
83.     scan.close();
84. }
85. }
```

Output:

advertisement

```
$ javac FloydWarshallShortestPath.java
$ java FloydWarshallShortestPath
```

Enter the number of vertices

6

Enter the Weighted Matrix for the graph

```
0 4 0 0 1 0
0 0 1 0 2 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 5 0 3
0 0 0 0 0 0
```

The Transitive Closure of the Graph (999 represents not reachable)

	1	2	3	4	5	6
1	0	4	5	6	1	4
2	999	0	1	7	2	5
3	999	999	0	999	999	999
4	999	999	999	0	999	999
5	999	999	999	5	0	3
6	999	999	999	999	999	0

322. Java Program to Find Shortest Path Between All Vertices Using Floyd-Warshall's Algorithm

[« Prev](#)

[Next »](#)

This is a java program to find shortest path between all vertices using FLoyd-Warshall's algorithm. In computer science, the Floyd–Warshall algorithm (also known as Floyd's algorithm, Roy–Warshall algorithm, Roy–Floyd algorithm, or the WFI algorithm) is a graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles, see below) and also for finding transitive closure of a relation R. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pairs of vertices, though it does not return details of the paths themselves.

Here is the source code of the Java Program to Find Shortest Path Between All Vertices Using Floyd-Warshall's Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.graph;
3. 
4. import java.util.Scanner;
5. 
6. public class FloydWarshallShortestPath
7. {
8.     private int      distancematrix[][];
9.     private int      numberofvertices;
10.    public static final int INFINITY = 999;
11. 
12.   public FloydWarshallShortestPath(int numberofvertices)
13.   {
14.       distancematrix = new int[numberofvertices + 1][numberofvertices + 1];
15.       this.numberofvertices = numberofvertices;
16.   }
17. 
18.   public void floydwarshall(int adjacencymatrix[][])
19.   {
20.       for (int source = 1; source <= numberofvertices; source++)
21.       {
22.           for (int destination = 1; destination <= numberofvertices; destination++)
23.           {
24.               distancematrix[source][destination] = adjacencymatrix[source][destination];
25.           }
26.       }
27.       for (int intermediate = 1; intermediate <= numberofvertices; intermediate++)
28.       {
29.           for (int source = 1; source <= numberofvertices; source++)
30.           {
31.               for (int destination = 1; destination <= numberofvertices; destination++)
32.               {
33.                   if (distancematrix[source][intermediate]
34.                       + distancematrix[intermediate][destination] < distancematrix[source][destination])
35.                       distancematrix[source][destination] = distancematrix[source][intermediate]
36.                                     + distancematrix[intermediate][destination];
37.               }
38.           }
39.       }
40.       for (int source = 1; source <= numberofvertices; source++)
41.           System.out.print("\t" + source);
42.       System.out.println();
43.       for (int source = 1; source <= numberofvertices; source++)
44.       {
45.           System.out.print(source + "\t");
```

```

46.     for (int destination = 1; destination <= numberofvertices; destination++)
47.     {
48.         System.out.print(distancematrix[source][destination] + "\t");
49.     }
50.     System.out.println();
51. }
52. }
53.
54. public static void main(String... arg)
55. {
56.     int adjacency_matrix[][];
57.     int numberofvertices;
58.     Scanner scan = new Scanner(System.in);
59.     System.out.println("Enter the number of vertices");
60.     numberofvertices = scan.nextInt();
61.     adjacency_matrix = new int[numberofvertices + 1][numberofvertices + 1];
62.     System.out.println("Enter the Weighted Matrix for the graph");
63.     for (int source = 1; source <= numberofvertices; source++)
64.     {
65.         for (int destination = 1; destination <= numberofvertices; destination++)
66.         {
67.             adjacency_matrix[source][destination] = scan.nextInt();
68.             if (source == destination)
69.             {
70.                 adjacency_matrix[source][destination] = 0;
71.                 continue;
72.             }
73.             if (adjacency_matrix[source][destination] == 0)
74.             {
75.                 adjacency_matrix[source][destination] = INFINITY;
76.             }
77.         }
78.     }
79.     System.out.println("The Transitive Closure of the Graph");
80.     FloydWarshallShortestPath floydwarshall = new FloydWarshallShortestPath(
81.         numberofvertices);
82.     floydwarshall.floydwarshall(adjacency_matrix);
83.     scan.close();
84. }
85. }
```

Output:

advertisement

```
$ javac FloydWarshallShortestPath.java
$ java FloydWarshallShortestPath
```

Enter the number of vertices

6

Enter the Weighted Matrix for the graph

```
0 4 0 0 1 0
0 0 1 0 2 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 5 0 3
0 0 0 0 0 0
```

The Transitive Closure of the Graph (999 represents not reachable)

	1	2	3	4	5	6
1	0	4	5	6	1	4
2	999	0	1	7	2	5
3	999	999	0	999	999	999
4	999	999	999	0	999	999
5	999	999	999	5	0	3
6	999	999	999	999	999	0

323. Java Program to Use Boruvka's Algorithm to Find the Minimum Spanning Tree

[« Prev](#)

[Next »](#)

This is a java program to find the minimum spanning tree of a graph. Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a weight to each edge, which is a number representing how unfavorable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A minimum spanning tree (MST) or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of minimum spanning trees for its connected components.

Here is the source code of the Java Program to Use Boruvka's Algorithm to Find the Minimum Spanning Tree. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.Iterator;
5. import java.util.NoSuchElementException;
6. import java.util.Scanner;
7.
8. public class BoruvkasMST
9. {
10.     private Bag<Edge> mst = new Bag<Edge>(); // Edge in MST
11.     private double weight; // weight of MST
12.
13.     public BoruvkasMST(EdgeWeightedGraph G)
14.     {
15.         UF uf = new UF(G.V());
16.         // repeat at most log V times or until we have V-1 Edge
17.         for (int t = 1; t < G.V() && mst.size() < G.V() - 1; t = t + t)
18.         {
19.             // foreach tree in forest, find closest edge
20.             // if edge weights are equal, ties are broken in favor of first edge
21.             // in G.Edge()
22.             Edge[] closest = new Edge[G.V()];
23.             for (Edge e : G.Edge())
24.             {
25.                 int v = e.either(), w = e.other(v);
26.                 int i = uf.find(v), j = uf.find(w);
27.                 if (i == j)
28.                     continue; // same tree
29.                 if (closest[i] == null || less(e, closest[i]))
30.                     closest[i] = e;
31.                 if (closest[j] == null || less(e, closest[j]))
32.                     closest[j] = e;
33.             }
34.             // add newly discovered Edge to MST
35.             for (int i = 0; i < G.V(); i++)
36.             {
37.                 Edge e = closest[i];
38.                 if (e != null)
39.                 {
40.                     int v = e.either(), w = e.other(v);
41.                     // don't add the same edge twice
42.                     if (!uf.connected(v, w))
43.                     {
```

```

44.         mst.add(e);
45.         weight += e.weight();
46.         uf.union(v, w);
47.     }
48. }
49. }
50. }
51. // check optimality conditions
52. assert check(G);
53. }
54.
55. public Iterable<Edge> Edge()
56. {
57.     return mst;
58. }
59.
60. public double weight()
61. {
62.     return weight;
63. }
64.
65. // is the weight of edge e strictly less than that of edge f?
66. private static boolean less(Edge e, Edge f)
67. {
68.     return e.weight() < f.weight();
69. }
70.
71. // check optimality conditions (takes time proportional to E V lg* V)
72. private boolean check(EdgeWeightedGraph G)
73. {
74.     // check weight
75.     double totalWeight = 0.0;
76.     for (Edge e : Edge())
77.     {
78.         totalWeight += e.weight();
79.     }
80.     double EPSILON = 1E-12;
81.     if (Math.abs(totalWeight - weight()) > EPSILON)
82.     {
83.         System.err.printf(
84.             "Weight of Edge does not equal weight(): %f vs. %f\n",
85.             totalWeight, weight());
86.         return false;
87.     }
88.     // check that it is acyclic
89.     UF uf = new UF(G.V());
90.     for (Edge e : Edge())
91.     {
92.         int v = e.either(), w = e.other(v);
93.         if (uf.connected(v, w))
94.         {
95.             System.err.println("Not a forest");
96.             return false;
97.         }
98.         uf.union(v, w);
99.     }
100.    // check that it is a spanning forest
101.    for (Edge e : G.Edge())
102.    {
103.        int v = e.either(), w = e.other(v);
104.        if (!uf.connected(v, w))
105.        {
106.            System.err.println("Not a spanning forest");
107.            return false;
108.        }
109.    }

```

```

110.    // check that it is a minimal spanning forest (cut optimality
111.    // conditions)
112.    for (Edge e : Edge())
113.    {
114.        // all Edge in MST except e
115.        uf = new UF(G.V());
116.        for (Edge f : mst)
117.        {
118.            int x = f.either(), y = f.other(x);
119.            if (f != e)
120.                uf.union(x, y);
121.        }
122.        // check that e is min weight edge in crossing cut
123.        for (Edge f : G.Edge())
124.        {
125.            int x = f.either(), y = f.other(x);
126.            if (!uf.connected(x, y))
127.            {
128.                if (f.weight() < e.weight())
129.                {
130.                    System.out.println("Edge " + f
131.                        + " violates cut optimality conditions");
132.                    return false;
133.                }
134.            }
135.        }
136.    }
137.    return true;
138. }
139.
140. public static void main(String[] args)
141. {
142.     Scanner sc = new Scanner(System.in);
143.     System.out.println("Enter the number of verties: ");
144.     EdgeWeightedGraph G = new EdgeWeightedGraph(sc.nextInt());
145.     BoruvkasMST mst = new BoruvkasMST(G);
146.     System.out.println("MST: ");
147.     for (Edge e : mst.Edge())
148.     {
149.         System.out.println(e);
150.     }
151.     System.out.printf("Total weight of MST: %.5f\n", mst.weight());
152.     sc.close();
153. }
154. }
155.
156. class BagOfItems<Item> implements Iterable<Item>
157. {
158.     private int N;           // number of elements in bag
159.     private Node<Item> first; // beginning of bag
160.
161.     // helper linked list class
162.     private static class Node<Item>
163.     {
164.         private Item item;
165.         private Node<Item> next;
166.     }
167.
168.     public BagOfItems()
169.     {
170.         first = null;
171.         N = 0;
172.     }
173.
174.     public boolean isEmpty()
175.     {

```

```

176.     return first == null;
177. }
178.
179. public int size()
180. {
181.     return N;
182. }
183.
184. public void add(Item item)
185. {
186.     Node<Item> oldfirst = first;
187.     first = new Node<Item>();
188.     first.item = item;
189.     first.next = oldfirst;
190.     N++;
191. }
192.
193. public Iterator<Item> iterator()
194. {
195.     return new ListIterator<Item>(first);
196. }
197.
198. // an iterator, doesn't implement remove() since it's optional
199. @SuppressWarnings("hiding")
200. private class ListIterator<Item> implements Iterator<Item>
201. {
202.     private Node<Item> current;
203.
204.     public ListIterator(Node<Item> first)
205.     {
206.         current = first;
207.     }
208.
209.     public boolean hasNext()
210.     {
211.         return current != null;
212.     }
213.
214.     public void remove()
215.     {
216.         throw new UnsupportedOperationException();
217.     }
218.
219.     public Item next()
220.     {
221.         if (!hasNext())
222.             throw new NoSuchElementException();
223.         Item item = current.item;
224.         current = current.next;
225.         return item;
226.     }
227. }
228. }
229.
230.class EdgeWeightedGraph
231.
232. private final int V;
233. private final int E;
234. private Bag<Edge>[] adj;
235.
236. @SuppressWarnings("unchecked")
237. public EdgeWeightedGraph(int V)
238. {
239.     Scanner sc = new Scanner(System.in);
240.     if (V < 0)
241.     {

```

```

242.     sc.close();
243.     throw new IllegalArgumentException(
244.         "Number of vertices must be nonnegative");
245. }
246. this.V = V;
247. adj = (Bag<Edge>[]) new Bag[V];
248. for (int v = 0; v < V; v++)
249. {
250.     adj[v] = new Bag<Edge>();
251. }
252. System.out.println("Enter the number of Edge: ");
253. E = sc.nextInt();
254. if (E < 0)
255. {
256.     sc.close();
257.     throw new IllegalArgumentException(
258.         "Number of Edge must be nonnegative");
259. }
260. System.out.println("Enter the Edge: <from> <to>");
261. for (int i = 0; i < E; i++)
262. {
263.     int v = sc.nextInt();
264.     int w = sc.nextInt();
265.     double weight = Math.round(100 * Math.random()) / 100.0;
266.     System.out.println(weight);
267.     Edge e = new Edge(v, w, weight);
268.     addEdge(e);
269. }
270. sc.close();
271. }
272.
273. public int V()
274. {
275.     return V;
276. }
277.
278. public int E()
279. {
280.     return E;
281. }
282.
283. public void addEdge(Edge e)
284. {
285.     int v = e.either();
286.     int w = e.other(v);
287.     if (v < 0 || v >= V)
288.         throw new IndexOutOfBoundsException("vertex " + v
289.             + " is not between 0 and " + (V - 1));
290.     if (w < 0 || w >= V)
291.         throw new IndexOutOfBoundsException("vertex " + w
292.             + " is not between 0 and " + (V - 1));
293.     adj[v].add(e);
294.     adj[w].add(e);
295. }
296.
297. public Iterable<Edge> adj(int v)
298. {
299.     if (v < 0 || v >= V)
300.         throw new IndexOutOfBoundsException("vertex " + v
301.             + " is not between 0 and " + (V - 1));
302.     return adj[v];
303. }
304.
305. public Iterable<Edge> Edge()
306. {
307.     Bag<Edge> list = new Bag<Edge>();

```

```

308.     for (int v = 0; v < V; v++)
309.     {
310.         int selfLoops = 0;
311.         for (Edge e : adj(v))
312.         {
313.             if (e.other(v) > v)
314.             {
315.                 list.add(e);
316.             }
317.             // only add one copy of each self loop (self loops will be
318.             // consecutive)
319.             else if (e.other(v) == v)
320.             {
321.                 if (selfLoops % 2 == 0)
322.                     list.add(e);
323.                 selfLoops++;
324.             }
325.         }
326.     }
327.     return list;
328. }
329.
330. public String toString()
331. {
332.     String NEWLINE = System.getProperty("line.separator");
333.     StringBuilder s = new StringBuilder();
334.     s.append(V + " " + E + NEWLINE);
335.     for (int v = 0; v < V; v++)
336.     {
337.         s.append(v + ": ");
338.         for (Edge e : adj[v])
339.         {
340.             s.append(e + " ");
341.         }
342.         s.append(NEWLINE);
343.     }
344.     return s.toString();
345. }
346.
347.
348.class Edge implements Comparable<Edge>
349. {
350.     private final int v;
351.     private final int w;
352.     private final double weight;
353.
354.     public Edge(int v, int w, double weight)
355.     {
356.         if (v < 0)
357.             throw new IndexOutOfBoundsException(
358.                 "Vertex name must be a nonnegative integer");
359.         if (w < 0)
360.             throw new IndexOutOfBoundsException(
361.                 "Vertex name must be a nonnegative integer");
362.         if (Double.isNaN(weight))
363.             throw new IllegalArgumentException("Weight is NaN");
364.         this.v = v;
365.         this.w = w;
366.         this.weight = weight;
367.     }
368.
369.     public double weight()
370.     {
371.         return weight;
372.     }
373.

```

```

374. public int either()
375. {
376.     return v;
377. }
378.
379. public int other(int vertex)
380. {
381.     if (vertex == v)
382.         return w;
383.     else if (vertex == w)
384.         return v;
385.     else
386.         throw new IllegalArgumentException("Illegal endpoint");
387. }
388.
389. public int compareTo(Edge that)
390. {
391.     if (this.weight() < that.weight())
392.         return -1;
393.     else if (this.weight() > that.weight())
394.         return +1;
395.     else
396.         return 0;
397. }
398.
399. public String toString()
400. {
401.     return String.format("%d-%d %.5f", v, w, weight);
402. }
403. }
404.
405.class UF
406. {
407.     private int[] id; // id[i] = parent of i
408.     private byte[] rank; // rank[i] = rank of subtree rooted at i (cannot be
409.                         // more than 31)
410.     private int count; // number of components
411.
412.     public UF(int N)
413.     {
414.         if (N < 0)
415.             throw new IllegalArgumentException();
416.         count = N;
417.         id = new int[N];
418.         rank = new byte[N];
419.         for (int i = 0; i < N; i++)
420.         {
421.             id[i] = i;
422.             rank[i] = 0;
423.         }
424.     }
425.
426.     public int find(int p)
427.     {
428.         if (p < 0 || p >= id.length)
429.             throw new IndexOutOfBoundsException();
430.         while (p != id[p])
431.         {
432.             id[p] = id[id[p]]; // path compression by halving
433.             p = id[p];
434.         }
435.         return p;
436.     }
437.
438.     public int count()
439.     {

```

```

440.     return count;
441. }
442.
443. public boolean connected(int p, int q)
444. {
445.     return find(p) == find(q);
446. }
447.
448. public void union(int p, int q)
449. {
450.     int i = find(p);
451.     int j = find(q);
452.     if (i == j)
453.         return;
454.     // make root of smaller rank point to root of larger rank
455.     if (rank[i] < rank[j])
456.         id[i] = j;
457.     else if (rank[i] > rank[j])
458.         id[j] = i;
459.     else
460.     {
461.         id[j] = i;
462.         rank[i]++;
463.     }
464.     count--;
465. }
466.

```

Output:

advertisement

```
$ javac BoruvkasMST.java
$ java BoruvkasMST
```

Enter the number of verties:

6

Enter the number of Edge:

7

Enter the Edge: <from> <to>

0 1

0.09

1 2

0.48

1 3

0.52

3 4

0.43

4 5

0.98

5 3

0.07

5 2

0.1

MST:

1-2 0.48000

3-4 0.43000

5-3 0.07000

5-2 0.10000

0-1 0.09000

Total weight of MST: 1.17000

324. Java Program to Create a Random Linear Extension for a DAG

[« Prev](#)

[Next »](#)

This is a java program to create a random linear extension of DAG. Linear extension of DAG is nothing but topological sorting in simple terms.

Here is the source code of the Java Program to Create a Random Linear Extension for a DAG. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6. import java.util.Stack;
7.
8. public class DAGLinearExtension
9. {
10.     private Stack<Integer> stack;
11.
12.     public DAGLinearExtension()
13.     {
14.         stack = new Stack<Integer>();
15.     }
16.
17.     public int[] topological(int adjacency_matrix[][], int source)
18.             throws NullPointerException
19.     {
20.         int number_of_nodes = adjacency_matrix[source].length - 1;
21.         int[] topological_sort = new int[number_of_nodes + 1];
22.         int pos = 1;
23.         int j;
24.         int visited[] = new int[number_of_nodes + 1];
25.         int element = source;
26.         int i = source;
27.         visited[source] = 1;
28.         stack.push(source);
29.         while (!stack.isEmpty())
30.         {
31.             element = stack.peek();
32.             while (i <= number_of_nodes)
33.             {
34.                 if (adjacency_matrix[element][i] == 1 && visited[i] == 1)
35.                 {
36.                     if (stack.contains(i))
37.                     {
38.                         System.out.println("TOPOLOGICAL SORT NOT POSSIBLE");
39.                         return null;
40.                     }
41.                 }
42.                 if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
43.                 {
44.                     stack.push(i);
45.                     visited[i] = 1;
46.                     element = i;
47.                     i = 1;
48.                     continue;
49.                 }
50.                 i++;
51.             }
52.         }
53.     }
54. }
```

```

52.         j = stack.pop();
53.         topological_sort[pos++] = j;
54.         i = ++j;
55.     }
56.     return topological_sort;
57. }
58.
59. public static void main(String... arg)
60. {
61.     int number_no_nodes, source;
62.     Scanner scanner = null;
63.     int topological_sort[] = null;
64.     System.out
65.         .println("Linear extension of a DAG is its topological representation.");
66.     try
67.     {
68.         System.out.println("Enter the number of nodes in the graph");
69.         scanner = new Scanner(System.in);
70.         number_no_nodes = scanner.nextInt();
71.         int adjacency_matrix[][] = new int[number_no_nodes + 1][number_no_nodes + 1];
72.         System.out.println("Enter the adjacency matrix");
73.         for (int i = 1; i <= number_no_nodes; i++)
74.             for (int j = 1; j <= number_no_nodes; j++)
75.                 adjacency_matrix[i][j] = scanner.nextInt();
76.         System.out.println("Enter the source for the graph");
77.         source = scanner.nextInt();
78.         System.out
79.             .println("The Topological sort for the graph is given by ");
80.         DAGLinearExtension toposort = new DAGLinearExtension();
81.         topological_sort = toposort.topological(adjacency_matrix, source);
82.         for (int i = topological_sort.length - 1; i > 0; i--)
83.         {
84.             if (topological_sort[i] != 0)
85.                 System.out.print(topological_sort[i] + "\t");
86.         }
87.     }
88.     catch (InputMismatchException inputMismatch)
89.     {
90.         System.out.println("Wrong Input format");
91.     }
92.     catch (NullPointerException nullPointer)
93.     {
94.     }
95.     scanner.close();
96. }
97.

```

Output:

```
$ javac DAGLinearExtension.java
$ java DAGLinearExtension
```

Linear extension of a DAG is its topological representation.

Enter the number of nodes in the graph

6

Enter the adjacency matrix

```
0 1 0 0 0 1
0 0 1 1 0 0
0 0 0 0 0 0
0 0 0 0 1 0
0 0 0 0 0 1
0 0 1 0 0 0
```

Enter the source for the graph

1

The Topological sort for the graph is given by

1	2	4	5	6	3
---	---	---	---	---	---

325. Java Program to Check Whether Topological Sorting can be Performed in a Graph

[« Prev](#)

[Next »](#)

This is a java program to check if topological sorting can be performed on graph or not. Topological sort exists only if there is not cycle in directed graph. In short this problem can be reduced to check if the given graph is Directed Acyclic Graph. If it is topological sort can be performed, not otherwise.

Here is the source code of the Java Program to Check Whether Topological Sorting can be Performed in a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.graph;
3. 
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10. 
11.class GraphLinkedList
12.{
13.    private Map<Integer, List<Integer>> adjacencyList;
14. 
15.    public GraphLinkedList(int v)
16.    {
17.        adjacencyList = new HashMap<Integer, List<Integer>>();
18.        for (int i = 1; i <= v; i++)
19.            adjacencyList.put(i, new LinkedList<Integer>());
20.    }
21. 
22.    public void setEdge(int from, int to)
23.    {
24.        if (to > adjacencyList.size() || from > adjacencyList.size())
25.            System.out.println("The vertices does not exists");
26.        /*
27.         * List<Integer> sls = adjacencyList.get(to);
28.         * sls.add(from);
29.         */
30.        List<Integer> dls = adjacencyList.get(from);
31.        dls.add(to);
32.    }
33. 
34.    public List<Integer> getEdge(int to)
35.    {
36.        if (to > adjacencyList.size())
37.        {
38.            System.out.println("The vertices does not exists");
39.            return null;
40.        }
41.        return adjacencyList.get(to);
42.    }
43. 
44.    public boolean checkDAG()
45.    {
46.        Integer count = 0;
47.        Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
48.        Integer size = this.adjacencyList.size() - 1;
49.        while (iteratorI.hasNext())
```

```

50.    {
51.        Integer i = iteratorI.next();
52.        List<Integer> adjList = this.adjacencyList.get(i);
53.        if (count == size)
54.        {
55.            return true;
56.        }
57.        if (adjList.size() == 0)
58.        {
59.            count++;
60.            System.out.println("Target Node - " + i);
61.            Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
62.                .iterator();
63.            while (iteratorJ.hasNext())
64.            {
65.                Integer j = iteratorJ.next();
66.                List<Integer> li = this.adjacencyList.get(j);
67.                if (li.contains(i))
68.                {
69.                    li.remove(i);
70.                    System.out.println("Deleting edge between target node "
71.                        + i + " - " + j + " ");
72.                }
73.            }
74.            this.adjacencyList.remove(i);
75.            iteratorI = this.adjacencyList.keySet().iterator();
76.        }
77.    }
78.    return false;
79. }
80.
81. public void printGraph()
82. {
83.     System.out.println("The Graph is: ");
84.     for (int i = 1; i <= this.adjacencyList.size(); i++)
85.     {
86.         List<Integer> edgeList = this.getEdge(i);
87.         if (edgeList.size() != 0)
88.         {
89.             System.out.print(i);
90.             for (int j = 0; j < edgeList.size(); j++)
91.             {
92.                 System.out.print(" -> " + edgeList.get(j));
93.             }
94.             System.out.println();
95.         }
96.     }
97. }
98.
99.
100. public class TopologicalSortPossible
101. {
102.     public static void main(String args[])
103.     {
104.         int v, e, count = 1, to, from;
105.         Scanner sc = new Scanner(System.in);
106.         GraphLinkedList glist;
107.         try
108.         {
109.             System.out.println("Enter the number of vertices: ");
110.             v = sc.nextInt();
111.             System.out.println("Enter the number of edges: ");
112.             e = sc.nextInt();
113.             glist = new GraphLinkedList(v);
114.             System.out.println("Enter the edges in the graph : <from> <to>");
115.             while (count <= e)

```

```

116.    {
117.        to = sc.nextInt();
118.        from = sc.nextInt();
119.        glist.setEdge(to, from);
120.        count++;
121.    }
122.    glist.printGraph();
123.    System.out
124.        .println("--Processing graph to check whether it is DAG--");
125.    if (glist.checkDAG())
126.    {
127.        System.out
128.            .println("Result: \nGiven graph is DAG (Directed Acyclic Graph).");
129.    }
130.    else
131.    {
132.        System.out
133.            .println("Result: \nGiven graph is not DAG (Directed Acyclic Graph).");
134.    }
135. }
136. catch (Exception E)
137. {
138.     System.out
139.         .println("You are trying to access empty adjacency list of a node.");
140. }
141. sc.close();
142. }
143.

```

Output:

advertisement

```
$ javac TopologicalSortPossible.java
$ java TopologicalSortPossible
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

5 6

6 1

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 1 -> 3

--Processing graph to check whether it is DAG--

Target Node - 3

Deleting edge between target node 3 - 2

Deleting edge between target node 3 - 6

Result:

Given graph is not DAG (Directed Acyclic Graph).

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3
2 4
4 5
4 6
5 6
6 3

The Graph is:

1 -> 2
2 -> 3 -> 4
4 -> 5 -> 6
5 -> 6
6 -> 3

--Processing graph to check whether it is DAG--

Target Node - 3

Deleting edge between target node 3 - 2

Deleting edge between target node 3 - 6

Target Node - 6

Deleting edge between target node 6 - 4

Deleting edge between target node 6 - 5

Target Node - 5

Deleting edge between target node 5 - 4

Target Node - 4

Deleting edge between target node 4 - 2

Target Node - 2

Deleting edge between target node 2 - 1

Result:

Given graph is DAG (Directed Acyclic Graph).

326. Java Program to Apply DFS to Perform the Topological Sorting of a Directed Acyclic Graph

[« Prev](#)

[Next »](#)

This is a java program to find topological sort of DAG. In computer science, a topological sort (sometimes abbreviated topsort or toposort) or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v , u comes before v in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another; in this application, a topological ordering is just a valid sequence for the tasks. A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering, and algorithms are known for constructing a topological ordering of any DAG in linear time.

Here is the source code of the Java Program to Apply DFS to Perform the Topological Sorting of a Directed Acyclic Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6. import java.util.Stack;
7.
8. public class DigraphTopologicalSortingDFS
9. {
10.     private Stack<Integer> stack;
11.
12.     public DigraphTopologicalSortingDFS()
13.     {
14.         stack = new Stack<Integer>();
15.     }
16.
17.     public int[] topological(int adjacency_matrix[][], int source)
18.             throws NullPointerException
19.     {
20.         int number_of_nodes = adjacency_matrix[source].length - 1;
21.         int[] topological_sort = new int[number_of_nodes + 1];
22.         int pos = 1;
23.         int j;
24.         int visited[] = new int[number_of_nodes + 1];
25.         int element = source;
26.         int i = source;
27.         visited[source] = 1;
28.         stack.push(source);
29.         while (!stack.isEmpty())
30.         {
31.             element = stack.peek();
32.             while (i <= number_of_nodes)
33.             {
34.                 if (adjacency_matrix[element][i] == 1 && visited[i] == 1)
35.                 {
36.                     if (stack.contains(i))
37.                     {
38.                         System.out.println("TOPOLOGICAL SORT NOT POSSIBLE");
39.                         return null;
40.                     }
41.                 }
42.                 if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
43.                 {
```

```

44.         stack.push(i);
45.         visited[i] = 1;
46.         element = i;
47.         i = 1;
48.         continue;
49.     }
50.     i++;
51. }
52. j = stack.pop();
53. topological_sort[pos++] = j;
54. i = ++j;
55. }
56. return topological_sort;
57. }
58.
59. public static void main(String... arg)
60. {
61.     int number_no_nodes, source;
62.     Scanner scanner = null;
63.     int topological_sort[] = null;
64.     try
65.     {
66.         System.out.println("Enter the number of nodes in the graph");
67.         scanner = new Scanner(System.in);
68.         number_no_nodes = scanner.nextInt();
69.         int adjacency_matrix[][] = new int[number_no_nodes + 1][number_no_nodes + 1];
70.         System.out.println("Enter the adjacency matrix");
71.         for (int i = 1; i <= number_no_nodes; i++)
72.             for (int j = 1; j <= number_no_nodes; j++)
73.                 adjacency_matrix[i][j] = scanner.nextInt();
74.         System.out.println("Enter the source for the graph");
75.         source = scanner.nextInt();
76.         System.out
77.             .println("The Topological sort for the graph is given by ");
78.         DigraphTopologicalSortingDFS toposort = new DigraphTopologicalSortingDFS();
79.         topological_sort = toposort.topological(adjacency_matrix, source);
80.         for (int i = topological_sort.length - 1; i > 0; i--)
81.         {
82.             if (topological_sort[i] != 0)
83.                 System.out.print(topological_sort[i] + "\t");
84.         }
85.     }
86.     catch (InputMismatchException inputMismatch)
87.     {
88.         System.out.println("Wrong Input format");
89.     }
90.     catch (NullPointerException nullPointer)
91.     {
92.     }
93.     scanner.close();
94. }
95. }

```

Output:

advertisement

Enter the number of nodes in the graph

6

Enter the adjacency matrix

```

0 1 0 0 0 0
0 0 1 1 0 0
0 0 0 0 0 0
0 0 0 0 1 0
0 0 0 0 0 1

```

0 0 1 1 0 0

Enter the source for the graph

1

The Topological sort for the graph is given by

TOPOLOGICAL SORT NOT POSSIBLE

Enter the number of nodes in the graph

6

Enter the adjacency matrix

0 1 0 0 0 1

0 0 1 1 0 0

0 0 0 0 0 0

0 0 0 0 1 0

0 0 0 0 0 1

0 0 1 0 0 0

Enter the source for the graph

1

The Topological sort for the graph is given by

1 2 4 5 6 3

327.Java Program to Create a Minimal Set of All Edges Whose Addition will Convert it to a Strongly Connected DAG

[« Prev](#)

[Next »](#)

This is a java program to find the edges other than feedback arc set so that all the edges contribute to directed acyclic graph.

Here is the source code of the Java Program to Create a Minimal Set of All Edges Whose Addition will Convert it to a Strongly Connected DAG. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.graph;
3. 
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10. 
11.class Graph
12.{
13.    private Map<Integer, List<Integer>> adjacencyList;
14. 
15.    public Graph(int v)
16.    {
17.        adjacencyList = new HashMap<Integer, List<Integer>>();
18.        for (int i = 1; i <= v; i++)
19.            adjacencyList.put(i, new LinkedList<Integer>());
20.    }
21. 
22.    public void setEdge(int from, int to)
23.    {
24.        if (to > adjacencyList.size() || from > adjacencyList.size())
25.            System.out.println("The vertices does not exists");
26.        /*
27.         * List<Integer> sls = adjacencyList.get(to);
28.         * sls.add(from);
29.         */
30.        List<Integer> dls = adjacencyList.get(from);
31.        dls.add(to);
32.    }
33. 
34.    public List<Integer> getEdge(int to)
35.    {
36.        /*
37.         * if (to > adjacencyList.size())
38.         * {
39.         *     System.out.println("The vertices does not exists");
40.         *     return null;
41.         * }
42.         */
43.        return adjacencyList.get(to);
44.    }
45. 
46.    public Graph checkDAG()
47.    {
48.        Integer count = 0;
```

```

49.     Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
50.     Integer size = this.adjacencyList.size() - 1;
51.     System.out.println("Minimal set of edges: ");
52.     while (iteratorI.hasNext())
53.     {
54.         Integer i = iteratorI.next();
55.         List<Integer> adjList = this.adjacencyList.get(i);
56.         if (count == size)
57.         {
58.             return this;
59.         }
60.         if (adjList.size() == 0)
61.         {
62.             count++;
63.             Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
64.                 .iterator();
65.             while (iteratorJ.hasNext())
66.             {
67.                 Integer j = iteratorJ.next();
68.                 List<Integer> li = this.adjacencyList.get(j);
69.                 if (li.contains(i))
70.                 {
71.                     li.remove(i);
72.                     System.out.println(i + " -> " + j);
73.                 }
74.             }
75.             this.adjacencyList.remove(i);
76.             iteratorI = this.adjacencyList.keySet().iterator();
77.         }
78.     }
79.     return this;
80. }
81.
82. public Map<Integer, List<Integer>> getFeedbackArcSet(int v)
83. {
84.     int[] visited = new int[v + 1];
85.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
86.     Map<Integer, List<Integer>> l = new HashMap<Integer, List<Integer>>();
87.     while (iterator.hasNext())
88.     {
89.         Integer i = iterator.next();
90.         List<Integer> list = this.adjacencyList.get(i);
91.         visited[i] = 1;
92.         if (list.size() != 0)
93.         {
94.             for (int j = 0; j < list.size(); j++)
95.             {
96.                 if (visited[list.get(j)] == 1)
97.                 {
98.                     l.put(i, new LinkedList<Integer>());
99.                     l.get(i).add(j);
100.                }
101.                else
102.                {
103.                    visited[list.get(j)] = 1;
104.                }
105.            }
106.        }
107.    }
108.    return l;
109. }
110.
111. public void printAllEdges(Graph copyG, int v)
112. {
113.     Map<Integer, List<Integer>> edges = this.getFeedbackArcSet(v);
114.     Iterator<Integer> iterator = copyG.adjacencyList.keySet().iterator();

```

```

115.     while (iterator.hasNext())
116.     {
117.         Integer i = iterator.next();
118.         List<Integer> edgeList = this.getEdge(i);
119.         if (edgeList.size() != 0)
120.         {
121.             for (int j = 0; j < edgeList.size(); j++)
122.             {
123.                 if (edges.containsKey(i) && edges.get(i).contains(j))
124.                     continue;
125.                 else
126.                 {
127.                     System.out.print(i + " -> " + edgeList.get(j));
128.                 }
129.             }
130.             System.out.println();
131.         }
132.     }
133. }
134.
135. public void printGraph()
136. {
137.     System.out.println("The Graph is:");
138.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
139.     while (iterator.hasNext())
140.     {
141.         Integer i = iterator.next();
142.         List<Integer> edgeList = this.getEdge(i);
143.         if (edgeList.size() != 0)
144.         {
145.             System.out.print(i);
146.             for (int j = 0; j < edgeList.size(); j++)
147.             {
148.                 System.out.print(" -> " + edgeList.get(j));
149.             }
150.             System.out.println();
151.         }
152.     }
153. }
154. }
155.
156. public class MinimalSetofEdgesforDAG
157. {
158.     public static void main(String args[])
159.     {
160.         int v, e, count = 1, to, from;
161.         Scanner sc = new Scanner(System.in);
162.         Graph glist;
163.         try
164.         {
165.             System.out.println("Enter the number of vertices: ");
166.             v = sc.nextInt();
167.             System.out.println("Enter the number of edges: ");
168.             e = sc.nextInt();
169.             glist = new Graph(v);
170.             System.out.println("Enter the edges in the graph : <from> <to>");
171.             while (count <= e)
172.             {
173.                 to = sc.nextInt();
174.                 from = sc.nextInt();
175.                 glist.setEdge(to, from);
176.                 count++;
177.             }
178.             Graph copyofGlist = new Graph(v);
179.             copyofGlist = glist;
180.             glist.printGraph();

```

```
181.     Graph modified = glist.checkDAG();
182.     modified.printAllEdges(copyofGlist, v);
183. }
184. catch (Exception E)
185. {
186.     System.out
187.         .println("You are trying to access empty adjacency list of a node.");
188. }
189. sc.close();
190. }
191. }
```

Output:

advertisement

```
$ javac MinimalSetofEdgesforDAG.java
$ java MinimalSetofEdgesforDAG
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

5 6

6 3

6 4

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 3 -> 4

Minimal set of edges:

3 -> 2

3 -> 6

1 -> 2

2 -> 4

4 -> 5

5 -> 6

328.Java Program to Find the Connected Components of an UnDirected Graph

[« Prev](#)

[Next »](#)

This is a java program In graph theory, a connected component (or just component) of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph. A graph that is itself connected has exactly one connected component, consisting of the whole graph.

Here is the source code of the Java Program to Find the Connected Components of an UnDirected Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. // Sample program to find connected components of undirected graph
3. package com.sanfoundry.graph;
4.
5. import java.util.LinkedList;
6. import java.util.Queue;
7. import java.util.Scanner;
8.
9. class CCGraph
10. {
11.     static final int MAXV    = 100;
12.     static final int MAXDEGREE = 50;
13.     public int    edges[][] = new int[MAXV + 1][MAXDEGREE];
14.     public int    degree[] = new int[MAXV + 1];
15.     public int    nvertices;
16.     public int    nedges;
17.
18.     CCGraph()
19.     {
20.         nvertices = nedges = 0;
21.         for (int i = 1; i <= MAXV; i++)
22.             degree[i] = 0;
23.     }
24.
25.     void read_CCGraph(boolean directed)
26.     {
27.         int x, y;
28.         Scanner sc = new Scanner(System.in);
29.         System.out.println("Enter the number of vertices: ");
30.         nvertices = sc.nextInt();
31.         System.out.println("Enter the number of edges: ");
32.         int m = sc.nextInt();
33.         System.out.println("Enter the edges: <from> <to>");
34.         for (int i = 1; i <= m; i++)
35.         {
36.             x = sc.nextInt();
37.             y = sc.nextInt();
38.             insert_edge(x, y, directed);
39.         }
40.         sc.close();
41.     }
42.
43.     void insert_edge(int x, int y, boolean directed)
44.     {
45.         if (degree[x] > MAXDEGREE)
46.             System.out.printf(
47.                 "Warning: insertion (%d, %d) exceeds max degree\n", x, y);
48.         edges[x][degree[x]] = y;
49.         degree[x]++;
50.     }
51.
52.     void print_CCGraph()
53.     {
54.         for (int i = 1; i <= MAXV; i++)
55.             System.out.print(i + " : ");
56.             for (int j = 0; j < degree[i]; j++)
57.                 System.out.print(edges[i][j] + " ");
58.             System.out.println();
59.     }
60.
61.     void dfs(int v, boolean visited[])
62.     {
63.         visited[v] = true;
64.         System.out.print(v + " ");
65.         for (int i = 0; i < degree[v]; i++)
66.             if (!visited[edges[v][i]])
67.                 dfs(edges[v][i], visited);
68.     }
69.
70.     void print_CCComponents()
71.     {
72.         boolean visited[] = new boolean[MAXV + 1];
73.         for (int i = 1; i <= MAXV; i++)
74.             if (!visited[i])
75.                 dfs(i, visited);
76.     }
77.
78. }
```

```

50.    if (!directed)
51.        insert_edge(y, x, true);
52.    else
53.        nedges++;
54. }
55.
56. void print_CCGraph()
57. {
58.     for (int i = 1; i <= nvertices; i++)
59.     {
60.         System.out.printf("%d: ", i);
61.         for (int j = degree[i] - 1; j >= 0; j--)
62.             System.out.printf(" %d", edges[i][j]);
63.         System.out.printf("\n");
64.     }
65. }
66.
67.
68. public class ConnectedComponents
69. {
70.     static final int MAXV      = 100;
71.     static boolean processed[] = new boolean[MAXV];
72.     static boolean discovered[] = new boolean[MAXV];
73.     static int    parent[]    = new int[MAXV];
74.
75.     static void bfs(CCGraph g, int start)
76.     {
77.         Queue<Integer> q = new LinkedList<Integer>();
78.         int i, v;
79.         q.offer(start);
80.         discovered[start] = true;
81.         while (!q.isEmpty())
82.         {
83.             v = q.remove();
84.             process_vertex(v);
85.             processed[v] = true;
86.             for (i = g.degree[v] - 1; i >= 0; i--)
87.             {
88.                 if (!discovered[g.edges[v][i]])
89.                 {
90.                     q.offer(g.edges[v][i]);
91.                     discovered[g.edges[v][i]] = true;
92.                     parent[g.edges[v][i]] = v;
93.                 }
94.             }
95.         }
96.     }
97.
98.     static void initialize_search(CCGraph g)
99.     {
100.         for (int i = 1; i <= g.nvertices; i++)
101.         {
102.             processed[i] = discovered[i] = false;
103.             parent[i] = -1;
104.         }
105.     }
106.
107.     static void process_vertex(int v)
108.     {
109.         System.out.printf(" %d", v);
110.     }
111.
112.     static void connected_components(CCGraph g)
113.     {
114.         int c;
115.         initialize_search(g);

```

```

116.     c = 0;
117.     for (int i = 1; i <= g.nvertices; i++)
118.     {
119.         if (!discovered[i])
120.         {
121.             c++;
122.             System.out.printf("Component %d:", c);
123.             bfs(g, i);
124.             System.out.printf("\n");
125.         }
126.     }
127. }
128.
129. static public void main(String[] args)
130. {
131.     CCGraph g = new CCGraph();
132.     g.read_CCGraph(false);
133.     g.print_CCGraph();
134.     connected_components(g);
135. }
136.

```

Output:

advertisement

```
$ javac ConnectedComponents.java
$ java ConnectedComponents
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges: <from> <to>

1 2

2 3

2 4

4 5

5 6

6 3

6 4

1: 2

2: 4 3 1

3: 6 2

4: 6 5 2

5: 6 4

6: 4 3 5

Component 1: 1 2 4 3 6 5

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges: <from> <to>

1 2

1 4

1 3

2 3

5 6

6 5

4 3

1: 3 4 2

2: 3 1

3: 4 2 1

4: 3 1

5: 6 6

6: 5 5

Component 1: 1 3 4 2

Component 2: 5 6

329.Java Program to Check if a Directed Graph is a Tree or Not Using DFS

[« Prev](#)

[Next »](#)

This is a java program to check if graph is tree or not. Graph is tree if
1. It has number of edges one less than number of vertices.
2. Graph is connected.
3. There are no cycles.

Here is the source code of the Java Program to Check if a Directed Graph is a Tree or Not Using DFS. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.LinkedList;
5. import java.util.Queue;
6. import java.util.Scanner;
7.
8. class DTGraph
9. {
10.     static final int MAXV = 100;
11.     static final int MAXDEGREE = 50;
12.     public int edges[][] = new int[MAXV + 1][MAXDEGREE];
13.     public int degree[] = new int[MAXV + 1];
14.     public int nvertices;
15.     public int nedges;
16.
17.     DTGraph()
18.     {
19.         nvertices = nedges = 0;
20.         for (int i = 1; i <= MAXV; i++)
21.             degree[i] = 0;
22.     }
23.
24.     void read_CCGraph(boolean directed)
25.     {
26.         int x, y;
27.         Scanner sc = new Scanner(System.in);
28.         System.out.println("Enter the number of vertices: ");
29.         nvertices = sc.nextInt();
30.         System.out.println("Enter the number of edges: ");
31.         int m = sc.nextInt();
32.         System.out.println("Enter the edges: <from> <to>");
33.         for (int i = 1; i <= m; i++)
34.         {
35.             x = sc.nextInt();
36.             y = sc.nextInt();
37.             insert_edge(x, y, directed);
38.         }
39.         sc.close();
40.     }
41.
42.     void insert_edge(int x, int y, boolean directed)
43.     {
44.         if (degree[x] > MAXDEGREE)
45.             System.out.printf(
46.                 "Warning: insertion (%d, %d) exceeds max degree\n", x, y);
47.         edges[x][degree[x]] = y;
48.         degree[x]++;
49.         if (!directed)
```

```

50.     insert_edge(y, x, true);
51.   else
52.     nedges++;
53. }
54.
55. void print_CCGraph()
56. {
57.   for (int i = 1; i <= nvertices; i++)
58.   {
59.     System.out.printf("%d: ", i);
60.     for (int j = degree[i] - 1; j >= 0; j--)
61.       System.out.printf(" %d", edges[i][j]);
62.     System.out.printf("\n");
63.   }
64. }
65.}
66.
67.public class CheckDirectedGraphisTree
68.{
69.   static final int MAXV      = 100;
70.   static boolean processed[] = new boolean[MAXV];
71.   static boolean discovered[] = new boolean[MAXV];
72.   static int    parent[]    = new int[MAXV];
73.
74.   static void bfs(DTGraph g, int start)
75.   {
76.     Queue<Integer> q = new LinkedList<Integer>();
77.     int i, v;
78.     q.offer(start);
79.     discovered[start] = true;
80.     while (!q.isEmpty())
81.     {
82.       v = q.remove();
83.       // process_vertex(v);
84.       processed[v] = true;
85.       for (i = g.degree[v] - 1; i >= 0; i--)
86.       {
87.         if (!discovered[g.edges[v][i]])
88.         {
89.           q.offer(g.edges[v][i]);
90.           discovered[g.edges[v][i]] = true;
91.           parent[g.edges[v][i]] = v;
92.         }
93.       }
94.     }
95.   }
96.
97.   static void initialize_search(DTGraph g)
98.   {
99.     for (int i = 1; i <= g.nvertices; i++)
100.    {
101.      processed[i] = discovered[i] = false;
102.      parent[i] = -1;
103.    }
104.  }
105.
106.  static void process_vertex(int v)
107.  {
108.    System.out.printf(" %d", v);
109.  }
110.
111.  static int connected_components(DTGraph g)
112.  {
113.    int c;
114.    initialize_search(g);
115.    c = 0;

```

```

116.     for (int i = 1; i <= g.nvertices; i++)
117.     {
118.         if (!discovered[i])
119.         {
120.             c++;
121.             // System.out.printf("Component %d:", c);
122.             bfs(g, i);
123.             // System.out.printf("\n");
124.         }
125.     }
126.     return c;
127. }
128.
129. static public void main(String[] args)
130. {
131.     DTGraph g = new DTGraph();
132.     g.read_CCGraph(true);
133.     g.print_CCGraph();
134.     boolean flag = false;
135.     if (g.nedges == g.nvertices - 1)
136.     {
137.         flag = true;
138.         if (connected_components(g) == 1 && flag == true)
139.         {
140.             System.out
141.                 .println("Graph is a Tree, as graph is connected and Euler's criterion is satisfied.");
142.         }
143.     }
144.     else
145.     {
146.         System.out
147.             .println("Graph is not a Tree, as Euler's criterion is not satisfied.");
148.     }
149. }
150.

```

Output:

advertisement

```
$ javac CheckDirectedGraphisTree.java
$ java CheckDirectedGraphisTree
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges: <from> <to>

1 2

2 3

2 4

4 5

5 6

6 4

6 3

1: 2

2: 4 3

3:

4: 5

5: 6

6: 3 4

Graph is not a Tree, as Euler's criterion is not satisfied.

Enter the number of vertices:

4

Enter the number of edges:

3

Enter the edges: <from> <to>

1 3

1 2

3 4

1: 2 3

2:

3: 4

4:

Graph is a Tree, as graph is connected and Euler's criterion is satisfied.

330.Java Program to Check if an UnDirected Graph is a Tree or Not Using DFS

[« Prev](#)

[Next »](#)

This is a java program to check if graph is tree or not. Graph is tree if
1. It has number of edges one less than number of vertices.
2. Graph is connected.
3. There are no cycles.

Here is the source code of the Java Program to Check if an UnDirected Graph is a Tree or Not Using DFS. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.LinkedList;
5. import java.util.Queue;
6. import java.util.Scanner;
7.
8. class TGraph
9. {
10.     static final int MAXV = 100;
11.     static final int MAXDEGREE = 50;
12.     public int edges[][] = new int[MAXV + 1][MAXDEGREE];
13.     public int degree[] = new int[MAXV + 1];
14.     public int nvertices;
15.     public int nedges;
16.
17.     TGraph()
18.     {
19.         nvertices = nedges = 0;
20.         for (int i = 1; i <= MAXV; i++)
21.             degree[i] = 0;
22.     }
23.
24.     void read_CCGraph(boolean directed)
25.     {
26.         int x, y;
27.         Scanner sc = new Scanner(System.in);
28.         System.out.println("Enter the number of vertices: ");
29.         nvertices = sc.nextInt();
30.         System.out.println("Enter the number of edges: ");
31.         int m = sc.nextInt();
32.         System.out.println("Enter the edges: <from> <to>");
33.         for (int i = 1; i <= m; i++)
34.         {
35.             x = sc.nextInt();
36.             y = sc.nextInt();
37.             insert_edge(x, y, directed);
38.         }
39.         sc.close();
40.     }
41.
42.     void insert_edge(int x, int y, boolean directed)
43.     {
44.         if (degree[x] > MAXDEGREE)
45.             System.out.printf(
46.                 "Warning: insertion (%d, %d) exceeds max degree\n", x, y);
47.         edges[x][degree[x]] = y;
48.         degree[x]++;
49.         if (!directed)
```

```

50.     insert_edge(y, x, true);
51.   else
52.     nedges++;
53. }
54.
55. void print_CCGraph()
56. {
57.   for (int i = 1; i <= nvertices; i++)
58.   {
59.     System.out.printf("%d: ", i);
60.     for (int j = degree[i] - 1; j >= 0; j--)
61.       System.out.printf(" %d", edges[i][j]);
62.     System.out.printf("\n");
63.   }
64. }
65.}
66.
67.public class CheckUndirectedGraphisTree
68.{
69.  static final int MAXV      = 100;
70.  static boolean processed[] = new boolean[MAXV];
71.  static boolean discovered[] = new boolean[MAXV];
72.  static int    parent[]    = new int[MAXV];
73.
74.  static void bfs(TGraph g, int start)
75.  {
76.    Queue<Integer> q = new LinkedList<Integer>();
77.    int i, v;
78.    q.offer(start);
79.    discovered[start] = true;
80.    while (!q.isEmpty())
81.    {
82.      v = q.remove();
83.      // process_vertex(v);
84.      processed[v] = true;
85.      for (i = g.degree[v] - 1; i >= 0; i--)
86.      {
87.        if (!discovered[g.edges[v][i]])
88.        {
89.          q.offer(g.edges[v][i]);
90.          discovered[g.edges[v][i]] = true;
91.          parent[g.edges[v][i]] = v;
92.        }
93.      }
94.    }
95.  }
96.
97.  static void initialize_search(TGraph g)
98.  {
99.    for (int i = 1; i <= g.nvertices; i++)
100.    {
101.      processed[i] = discovered[i] = false;
102.      parent[i] = -1;
103.    }
104.  }
105.
106. static void process_vertex(int v)
107. {
108.   System.out.printf(" %d", v);
109. }
110.
111. static int connected_components(TGraph g)
112. {
113.   int c;
114.   initialize_search(g);
115.   c = 0;

```

```

116.     for (int i = 1; i <= g.nvertices; i++)
117.     {
118.         if (!discovered[i])
119.         {
120.             c++;
121.             // System.out.printf("Component %d:", c);
122.             bfs(g, i);
123.             // System.out.printf("\n");
124.         }
125.     }
126.     return c;
127. }
128.
129. static public void main(String[] args)
130. {
131.     TGraph g = new TGraph();
132.     g.read_CCGraph(false);
133.     g.print_CCGraph();
134.     boolean flag = false;
135.     if (g.nedges == g.nvertices - 1)
136.     {
137.         flag = true;
138.         if (connected_components(g) == 1 && flag == true)
139.         {
140.             System.out
141.                 .println("Graph is a Tree, as graph is connected and Euler's criterion is satisfied.");
142.         }
143.     }
144.     else
145.     {
146.         System.out
147.             .println("Graph is not a Tree, as Euler's criterion is not satisfied");
148.     }
149. }
150.

```

Output:

advertisement

```
$ javac CheckUndirectedGraphisTree.java
$ java CheckUndirectedGraphisTree
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges: <from> <to>

1 2

2 3

2 4

4 5

5 6

6 4

6 3

1: 2

2: 4 3 1

3: 6 2

4: 6 5 2

5: 6 4

6: 3 4 5

Graph is not a Tree, as Euler's criterion is not satisfied

Enter the number of vertices:

4

Enter the number of edges:

3

Enter the edges: <from> <to>

1 2
1 3
2 4
1: 3 2
2: 4 1
3: 1
4: 2

Graph is a Tree, as graph is connected and Euler's criterion is satisfied.

331.Java Program to Test Using DFS Whether a Directed Graph is Strongly Connected or Not

[« Prev](#)

[Next »](#)

This is a java program to test whether a directed graph is strongly connected or not. The graph is strongly connected if it has only one connected component.

Here is the source code of the Java Program to Test Using DFS Whether a Directed Graph is Strongly Connected or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.*;
5.
6. public class StronglyConnectedGraph
7. {
8.     private int          V;
9.     private int          preCount;
10.    private int[]         low;
11.    private boolean[]    visited;
12.    private List<Integer>[] graph;
13.    private List<List<Integer>> sccComp;
14.    private Stack<Integer>   stack;
15.
16.    /** function to get all strongly connected components */
17.    public List<List<Integer>> getSCComponents(List<Integer>[] graph)
18.    {
19.        V = graph.length;
20.        this.graph = graph;
21.        low = new int[V];
22.        visited = new boolean[V];
23.        stack = new Stack<Integer>();
24.        sccComp = new ArrayList<>();
25.        for (int v = 0; v < V; v++)
26.            if (!visited[v])
27.                dfs(v);
28.        return sccComp;
29.    }
30.
31.    /** function dfs */
32.    public void dfs(int v)
33.    {
34.        low[v] = preCount++;
35.        visited[v] = true;
36.        stack.push(v);
37.        int min = low[v];
38.        for (int w : graph[v])
39.        {
40.            if (!visited[w])
41.                dfs(w);
42.            if (low[w] < min)
43.                min = low[w];
44.        }
45.        if (min < low[v])
46.        {
47.            low[v] = min;
48.            return;
49.        }
50.        List<Integer> component = new ArrayList<Integer>();
```

```

51.     int w;
52.     do
53.     {
54.         w = stack.pop();
55.         component.add(w);
56.         low[w] = V;
57.     }
58.     while (w != v);
59.     sccComp.add(component);
60. }
61.
62. @SuppressWarnings("unchecked")
63. public static void main(String[] args)
64. {
65.     Scanner scan = new Scanner(System.in);
66.     System.out.println("Enter number of Vertices");
67.     /** number of vertices */
68.     int V = scan.nextInt();
69.     /** make graph */
70.     List<Integer>[] g = new List[V];
71.     for (int i = 0; i < V; i++)
72.         g[i] = new ArrayList<Integer>();
73.     /** accept all edges */
74.     System.out.println("Enter number of edges");
75.     int E = scan.nextInt();
76.     /** all edges */
77.     System.out.println("Enter the edges in the graph : <from> <to>");
78.     for (int i = 0; i < E; i++)
79.     {
80.         int x = scan.nextInt();
81.         int y = scan.nextInt();
82.         g[x].add(y);
83.     }
84.     StronglyConnectedGraph t = new StronglyConnectedGraph();
85.     System.out.print("The graph is strongly connected? : ");
86.     /** print all strongly connected components */
87.     List<List<Integer>> scComponents = t.getSCComponents(g);
88.     Iterator<List<Integer>> iterator = scComponents.iterator();
89.     boolean stronglyConnected = true;
90.     while (iterator.hasNext())
91.     {
92.         if (iterator.next().size() <= 1)
93.         {
94.             stronglyConnected = false;
95.         }
96.     }
97.     System.out.println(stronglyConnected);
98.     scan.close();
99. }
100.

```

Output:

advertisement

```
$ javac StronglyConnectedGraph.java
$ java StronglyConnectedGraph
```

```
Enter number of Vertices
6
Enter number of edges
7
Enter the edges in the graph : <from> <to>
0 1
1 2
1 3
3 4
```

4 5

5 3

5 2

The graph is strongly connected? : false

Enter number of Vertices

8

Enter number of edges

14

Enter the edges in the graph : <from> <to>

0 1

1 2

2 3

3 2

3 7

7 3

2 6

7 6

5 6

6 5

1 5

4 5

4 0

1 4

The graph is strongly connected? : true

332.Java Program to Test Using DFS Whether a Directed Graph is Weakly Connected or Not

[« Prev](#)

[Next »](#)

This is a java program to test whether a directed graph is weakly connected or not. The graph is weakly connected if it has more than one connected component.

Here is the source code of the Java Program to Test Using DFS Whether a Directed Graph is Weakly Connected or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.*;
5.
6. public class WeaklyConnectedGraph
7. {
8.     private int          V;
9.     private int          preCount;
10.    private int[]         low;
11.    private boolean[]    visited;
12.    private List<Integer>[] graph;
13.    private List<List<Integer>> sccComp;
14.    private Stack<Integer> stack;
15.
16.    /** function to get all strongly connected components */
17.    public List<List<Integer>> getSCComponents(List<Integer>[] graph)
18.    {
19.        V = graph.length;
20.        this.graph = graph;
21.        low = new int[V];
22.        visited = new boolean[V];
23.        stack = new Stack<Integer>();
24.        sccComp = new ArrayList<>();
25.        for (int v = 0; v < V; v++)
26.            if (!visited[v])
27.                dfs(v);
28.        return sccComp;
29.    }
30.
31.    /** function dfs */
32.    public void dfs(int v)
33.    {
34.        low[v] = preCount++;
35.        visited[v] = true;
36.        stack.push(v);
37.        int min = low[v];
38.        for (int w : graph[v])
39.        {
40.            if (!visited[w])
41.                dfs(w);
42.            if (low[w] < min)
43.                min = low[w];
44.        }
45.        if (min < low[v])
46.        {
47.            low[v] = min;
48.            return;
49.        }
50.        List<Integer> component = new ArrayList<Integer>();
```

```

51.    int w;
52.    do
53.    {
54.        w = stack.pop();
55.        component.add(w);
56.        low[w] = V;
57.    }
58.    while (w != v);
59.    sccComp.add(component);
60. }
61.
62. @SuppressWarnings("unchecked")
63. public static void main(String[] args)
64. {
65.     Scanner scan = new Scanner(System.in);
66.     System.out.println("Enter number of Vertices");
67.     /** number of vertices **/
68.     int V = scan.nextInt();
69.     /** make graph **/
70.     List<Integer>[] g = new List[V];
71.     for (int i = 0; i < V; i++)
72.         g[i] = new ArrayList<Integer>();
73.     /** accept all edges **/
74.     System.out.println("Enter number of edges");
75.     int E = scan.nextInt();
76.     /** all edges **/
77.     System.out.println("Enter the edges in the graph : <from> <to>");
78.     for (int i = 0; i < E; i++)
79.     {
80.         int x = scan.nextInt();
81.         int y = scan.nextInt();
82.         g[x].add(y);
83.     }
84.     StronglyConnectedGraph t = new StronglyConnectedGraph();
85.     System.out.print("The graph is weakly connected? : ");
86.     /** print all strongly connected components **/
87.     List<List<Integer>> scComponents = t.getSCComponents(g);
88.     Iterator<List<Integer>> iterator = scComponents.iterator();
89.     boolean weaklyConnected = false;
90.     while (iterator.hasNext())
91.     {
92.         if (iterator.next().size() <= 1)
93.         {
94.             weaklyConnected = true;
95.         }
96.     }
97.     System.out.println(weaklyConnected);
98.     scan.close();
99. }
100.

```

Output:

advertisement

```
$ javac WeaklyConnectedGraph.java
$ java WeaklyConnectedGraph
```

```
Enter number of Vertices
6
Enter number of edges
7
Enter the edges in the graph : <from> <to>
0 1
1 2
1 3
3 4
```

4 5

5 3

5 2

The graph is weakly connected? : true

333.Java Program to Remove the Edges in a Given Cyclic Graph such that its Linear Extension can be Found

[« Prev](#)

[Next »](#)

This is a java program to set of edges upon removal of which linear extension can be found. In simple terms this version of code finds the feedbackarc set, which when removed from graph leads to DAG for which we can find the topological sorting.

Here is the source code of the Java Program to Remove the Edges in a Given Cyclic Graph such that its Linear Extension can be Found. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.graph;
3.
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10.
11.class GraphLE
12.
13. private Map<Integer, List<Integer>> adjacencyList;
14.
15. public GraphLE(int v)
16. {
17.     adjacencyList = new HashMap<Integer, List<Integer>>();
18.     for (int i = 1; i <= v; i++)
19.         adjacencyList.put(i, new LinkedList<Integer>());
20. }
21.
22. public void setEdge(int from, int to)
23. {
24.     if (to > adjacencyList.size() || from > adjacencyList.size())
25.         System.out.println("The vertices does not exists");
26.     /*
27.      * List<Integer> sls = adjacencyList.get(to);
28.      * sls.add(from);
29.      */
30.     List<Integer> dls = adjacencyList.get(from);
31.     dls.add(to);
32. }
33.
34. public List<Integer> getEdge(int to)
35. {
36.     /*
37.      * if (to > adjacencyList.size())
38.      * {
39.      * System.out.println("The vertices does not exists");
40.      * return null;
41.      * }
42.      */
43.     return adjacencyList.get(to);
44. }
45.
46. public GraphLE checkDAG()
47. {
```

```

48.     Integer count = 0;
49.     Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
50.     Integer size = this.adjacencyList.size() - 1;
51.     while (iteratorI.hasNext())
52.     {
53.         Integer i = iteratorI.next();
54.         List<Integer> adjList = this.adjacencyList.get(i);
55.         if (count == size)
56.         {
57.             return this;
58.         }
59.         if (adjList.size() == 0)
60.         {
61.             count++;
62.             Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
63.                 .iterator();
64.             while (iteratorJ.hasNext())
65.             {
66.                 Integer j = iteratorJ.next();
67.                 List<Integer> li = this.adjacencyList.get(j);
68.                 if (li.contains(i))
69.                 {
70.                     li.remove(i);
71.                 }
72.             }
73.             this.adjacencyList.remove(i);
74.             iteratorI = this.adjacencyList.keySet().iterator();
75.         }
76.     }
77.     return this;
78. }
79.
80. public void printGraph()
81. {
82.     System.out.println("The Graph is: ");
83.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
84.     while (iterator.hasNext())
85.     {
86.         Integer i = iterator.next();
87.         List<Integer> edgeList = this.getEdge(i);
88.         if (edgeList.size() != 0)
89.         {
90.             System.out.print(i);
91.             for (int j = 0; j < edgeList.size(); j++)
92.             {
93.                 System.out.print(" -> " + edgeList.get(j));
94.             }
95.             System.out.println();
96.         }
97.     }
98. }
99.
100. public boolean removeEdgesToGetLinearExtension(int v)
101. {
102.     boolean flag = false;
103.     int[] visited = new int[v + 1];
104.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
105.     System.out.print("The set of edges in feedback arc set: ");
106.     while (iterator.hasNext())
107.     {
108.         Integer i = iterator.next();
109.         List<Integer> list = this.adjacencyList.get(i);
110.         visited[i] = 1;
111.         if (list.size() != 0)
112.         {
113.             for (int j = 0; j < list.size(); j++)

```

```

114.        {
115.            if (visited[list.get(j)] == 1)
116.            {
117.                flag = true;
118.                System.out.println(i + " - " + list.get(j));
119.            }
120.            else
121.            {
122.                visited[list.get(j)] = 1;
123.            }
124.        }
125.    }
126. }
127. return flag;
128. }
129.}
130.
131.public class RemoveEdgesLinearExtension
132.{
133.    public static void main(String args[])
134.    {
135.        int v, e, count = 1, to, from;
136.        Scanner sc = new Scanner(System.in);
137.        GraphLE glist;
138.        try
139.        {
140.            System.out.println("Enter the number of vertices: ");
141.            v = sc.nextInt();
142.            System.out.println("Enter the number of edges: ");
143.            e = sc.nextInt();
144.            glist = new GraphLE(v);
145.            System.out.println("Enter the edges in the graph : <from> <to>");
146.            while (count <= e)
147.            {
148.                to = sc.nextInt();
149.                from = sc.nextInt();
150.                glist.setEdge(to, from);
151.                count++;
152.            }
153.            glist.printGraph();
154.            GraphLE modified = glist.checkDAG();
155.            if (modified.removeEdgesToGetLinearExtension(v) == false)
156.            {
157.                System.out.println("None");
158.            }
159.        }
160.        catch (Exception E)
161.        {
162.            System.out
163.                .println("You are trying to access empty adjacency list of a node.");
164.        }
165.        sc.close();
166.    }
167.}

```

Output:

advertisement

```
$ javac RemoveEdgesLinearExtension.java
$ java RemoveEdgesLinearExtension
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

5 6

6 4

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 4 -> 3

The set of edges in feedback arc set: 6 - 4

334.Java Program to Find a Good Feedback Vertex Set

[« Prev](#)

[Next »](#)

This is a java program to find feedback vertex set. This is the set which contains vertices when removed from graph, graph becomes Directed acyclic graph.

Here is the source code of the Java Program to Find a Good Feedback Vertex Set. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10.
11.class GraphLL
12.
13. private Map<Integer, List<Integer>> adjacencyList;
14.
15. public GraphLL(int v)
16. {
17.     adjacencyList = new HashMap<Integer, List<Integer>>();
18.     for (int i = 1; i <= v; i++)
19.         adjacencyList.put(i, new LinkedList<Integer>());
20. }
21.
22. public void setEdge(int from, int to)
23. {
24.     if (to > adjacencyList.size() || from > adjacencyList.size())
25.         System.out.println("The vertices does not exists");
26.     /*
27.      * List<Integer> sls = adjacencyList.get(to);
28.      * sls.add(from);
29.      */
30.     List<Integer> dls = adjacencyList.get(from);
31.     dls.add(to);
32. }
33.
34. public List<Integer> getEdge(int to)
35. {
36.     /*
37.      * if (to > adjacencyList.size())
38.      * {
39.      *     System.out.println("The vertices does not exists");
40.      *     return null;
41.      * }
42.      */
43.     return adjacencyList.get(to);
44. }
45.
46. public GraphLL checkDAG()
47. {
48.     Integer count = 0;
49.     Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
50.     Integer size = this.adjacencyList.size() - 1;
51.     while (iteratorI.hasNext())
```

```

52.    {
53.        Integer i = iteratorI.next();
54.        List<Integer> adjList = this.adjacencyList.get(i);
55.        if (count == size)
56.        {
57.            return this;
58.        }
59.        if (adjList.size() == 0)
60.        {
61.            count++;
62.            Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
63.                .iterator();
64.            while (iteratorJ.hasNext())
65.            {
66.                Integer j = iteratorJ.next();
67.                List<Integer> li = this.adjacencyList.get(j);
68.                if (li.contains(i))
69.                {
70.                    li.remove(i);
71.                }
72.            }
73.            this.adjacencyList.remove(i);
74.            iteratorI = this.adjacencyList.keySet().iterator();
75.        }
76.    }
77.    return this;
78. }
79.
80. public void printGraph()
81. {
82.     System.out.println("The Graph is: ");
83.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
84.     while (iterator.hasNext())
85.     {
86.         Integer i = iterator.next();
87.         List<Integer> edgeList = this.getEdge(i);
88.         if (edgeList.size() != 0)
89.         {
90.             System.out.print(i);
91.             for (int j = 0; j < edgeList.size(); j++)
92.             {
93.                 System.out.print(" -> " + edgeList.get(j));
94.             }
95.             System.out.println();
96.         }
97.     }
98. }
99.
100. public boolean getFeedbackVertexSet(int v)
101. {
102.     boolean flag = false;
103.     int[] visited = new int[v + 1];
104.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
105.     System.out.print("The set of vertices in feedback vertex set: ");
106.     while (iterator.hasNext())
107.     {
108.         Integer i = iterator.next();
109.         List<Integer> list = this.adjacencyList.get(i);
110.         visited[i] = 1;
111.         if (list.size() != 0)
112.         {
113.             for (int j = 0; j < list.size(); j++)
114.             {
115.                 if (visited[list.get(j)] == 1)
116.                 {
117.                     flag = true;

```

```

118.         System.out.println(list.get(j) + " ");
119.     }
120.     else
121.     {
122.         visited[[list.get(j)]] = 1;
123.     }
124. }
125. }
126. }
127. return flag;
128. }
129. }
130.
131.public class FeedbackVertexSet
132. {
133.     public static void main(String args[])
134.     {
135.         int v, e, count = 1, to, from;
136.         Scanner sc = new Scanner(System.in);
137.         GraphLL glist;
138.         try
139.         {
140.             System.out.println("Enter the number of vertices: ");
141.             v = sc.nextInt();
142.             System.out.println("Enter the number of edges: ");
143.             e = sc.nextInt();
144.             glist = new GraphLL(v);
145.             System.out.println("Enter the edges in the graph : <from> <to>");
146.             while (count <= e)
147.             {
148.                 to = sc.nextInt();
149.                 from = sc.nextInt();
150.                 glist.setEdge(to, from);
151.                 count++;
152.             }
153.             glist.printGraph();
154.             GraphLL modified = glist.checkDAG();
155.             if (modified.getFeedbackVertexSet(v) == false)
156.             {
157.                 System.out.println("None");
158.             }
159.         }
160.         catch (Exception E)
161.         {
162.             System.out
163.                 .println("You are trying to access empty adjacency list of a node.");
164.         }
165.         sc.close();
166.     }
167. }

```

Output:

advertisement

```

Enter the number of vertices:
6
Enter the number of edges:
7
Enter the edges in the graph : <from> <to>
1 2
2 3
2 4
4 5
5 6

```

6 4

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 4 -> 3

The set of vertices in feedback vertex set: 4

335.Java Program to Find a Good Feedback Edge Set in a Graph

[« Prev](#)

[Next »](#)

This is a java program to find feednack arc set. This is the set which contains edges which when removed from the graph, graph becomes directed acyclic graph.

Here is the source code of the Java Program to Find a Good Feedback Edge Set in a Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10.
11.class Graph
12.
13. private Map<Integer, List<Integer>> adjacencyList;
14.
15. public Graph(int v)
16. {
17.     adjacencyList = new HashMap<Integer, List<Integer>>();
18.     for (int i = 1; i <= v; i++)
19.         adjacencyList.put(i, new LinkedList<Integer>());
20. }
21.
22. public void setEdge(int from, int to)
23. {
24.     if (to > adjacencyList.size() || from > adjacencyList.size())
25.         System.out.println("The vertices does not exists");
26.     /*
27.      * List<Integer> sls = adjacencyList.get(to);
28.      * sls.add(from);
29.      */
30.     List<Integer> dls = adjacencyList.get(from);
31.     dls.add(to);
32. }
33.
34. public List<Integer> getEdge(int to)
35. {
36.     /*
37.      * if (to > adjacencyList.size())
38.      * {
39.      *     System.out.println("The vertices does not exists");
40.      *     return null;
41.      * }
42.      */
43.     return adjacencyList.get(to);
44. }
45.
46. public Graph checkDAG()
47. {
48.     Integer count = 0;
49.     Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
50.     Integer size = this.adjacencyList.size() - 1;
51.     while (iteratorI.hasNext())
```

```

52.    {
53.        Integer i = iteratorI.next();
54.        List<Integer> adjList = this.adjacencyList.get(i);
55.        if (count == size)
56.        {
57.            return this;
58.        }
59.        if (adjList.size() == 0)
60.        {
61.            count++;
62.            Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
63.                .iterator();
64.            while (iteratorJ.hasNext())
65.            {
66.                Integer j = iteratorJ.next();
67.                List<Integer> li = this.adjacencyList.get(j);
68.                if (li.contains(i))
69.                {
70.                    li.remove(i);
71.                }
72.            }
73.            this.adjacencyList.remove(i);
74.            iteratorI = this.adjacencyList.keySet().iterator();
75.        }
76.    }
77.    return this;
78. }
79.
80. public void printGraph()
81. {
82.     System.out.println("The Graph is: ");
83.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
84.     while (iterator.hasNext())
85.     {
86.         Integer i = iterator.next();
87.         List<Integer> edgeList = this.getEdge(i);
88.         if (edgeList.size() != 0)
89.         {
90.             System.out.print(i);
91.             for (int j = 0; j < edgeList.size(); j++)
92.             {
93.                 System.out.print(" -> " + edgeList.get(j));
94.             }
95.             System.out.println();
96.         }
97.     }
98. }
99.
100. public boolean getFeedbackArcSet(int v)
101. {
102.     boolean flag = false;
103.     int[] visited = new int[v + 1];
104.     Iterator<Integer> iterator = this.adjacencyList.keySet().iterator();
105.     System.out.print("The set of edges in feedback arc set: ");
106.     while (iterator.hasNext())
107.     {
108.         Integer i = iterator.next();
109.         List<Integer> list = this.adjacencyList.get(i);
110.         visited[i] = 1;
111.         if (list.size() != 0)
112.         {
113.             for (int j = 0; j < list.size(); j++)
114.             {
115.                 if (visited[list.get(j)] == 1)
116.                 {
117.                     flag = true;

```

```

118.             System.out.println(i + " - " + list.get(j));
119.         }
120.     else
121.     {
122.         visited[list.get(j)] = 1;
123.     }
124. }
125. }
126. }
127. return flag;
128. }
129.}
130.

131.public class FeedbackArcSet
132.{
133.    public static void main(String args[])
134.    {
135.        int v, e, count = 1, to, from;
136.        Scanner sc = new Scanner(System.in);
137.        Graph glist;
138.        try
139.        {
140.            System.out.println("Enter the number of vertices: ");
141.            v = sc.nextInt();
142.            System.out.println("Enter the number of edges: ");
143.            e = sc.nextInt();
144.            glist = new Graph(v);
145.            System.out.println("Enter the edges in the graph : <from> <to>");
146.            while (count <= e)
147.            {
148.                to = sc.nextInt();
149.                from = sc.nextInt();
150.                glist.setEdge(to, from);
151.                count++;
152.            }
153.            glist.printGraph();
154.            Graph modified = glist.checkDAG();
155.            if (modified.getFeedbackArcSet(v) == false)
156.            {
157.                System.out.println("None");
158.            }
159.        }
160.        catch (Exception E)
161.        {
162.            System.out
163.                .println("You are trying to access empty adjacency list of a node.");
164.        }
165.        sc.close();
166.    }
167.}

```

Output:

advertisement

```
$ javac FeedbackArcSet.java
$ java FeedbackArcSet
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

5 6

6 4

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 4 -> 3

The set of edges in feedback arc set: 6 - 4

336.Java Program to Check Whether Graph is DAG

[« Prev](#)

[Next »](#)

This is a java program to check whether graph is DAG. In mathematics and computer science, a directed acyclic graph (DAG Listeni/'dæg/), is a directed graph with no directed cycles. That is, it is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is no way to start at some vertex v and follow a sequence of edges that eventually loops back to v again.

Here is the source code of the Java Program to Check Whether Graph is DAG. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.HashMap;
5. import java.util.Iterator;
6. import java.util.LinkedList;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10.
11.class GraphLinkedList
12.
13. private Map<Integer, List<Integer>> adjacencyList;
14.
15. public GraphLinkedList(int v)
16. {
17.     adjacencyList = new HashMap<Integer, List<Integer>>();
18.     for (int i = 1; i <= v; i++)
19.         adjacencyList.put(i, new LinkedList<Integer>());
20. }
21.
22. public void setEdge(int from, int to)
23. {
24.     if (to > adjacencyList.size() || from > adjacencyList.size())
25.         System.out.println("The vertices does not exists");
26.     /*
27.      * List<Integer> sls = adjacencyList.get(to);
28.      * sls.add(from);
29.      */
30.     List<Integer> dls = adjacencyList.get(from);
31.     dls.add(to);
32. }
33.
34. public List<Integer> getEdge(int to)
35. {
36.     if (to > adjacencyList.size())
37.     {
38.         System.out.println("The vertices does not exists");
39.         return null;
40.     }
41.     return adjacencyList.get(to);
42. }
43.
44. public boolean checkDAG()
45. {
46.     Integer count = 0;
47.     Iterator<Integer> iteratorI = this.adjacencyList.keySet().iterator();
48.     Integer size = this.adjacencyList.size() - 1;
49.     while (iteratorI.hasNext())
50.     {
51.         Integer i = iteratorI.next();
```

```

52.     List<Integer> adjList = this.adjacencyList.get(i);
53.     if (count == size)
54.     {
55.         return true;
56.     }
57.     if (adjList.size() == 0)
58.     {
59.         count++;
60.         System.out.println("Target Node - " + i);
61.         Iterator<Integer> iteratorJ = this.adjacencyList.keySet()
62.             .iterator();
63.         while (iteratorJ.hasNext())
64.         {
65.             Integer j = iteratorJ.next();
66.             List<Integer> li = this.adjacencyList.get(j);
67.             if (li.contains(i))
68.             {
69.                 li.remove(i);
70.                 System.out.println("Deleting edge between target node "
71.                     + i + " - " + j + " ");
72.             }
73.         }
74.         this.adjacencyList.remove(i);
75.         iteratorI = this.adjacencyList.keySet().iterator();
76.     }
77. }
78. return false;
79. }
80.
81. public void printGraph()
82. {
83.     System.out.println("The Graph is: ");
84.     for (int i = 1; i <= this.adjacencyList.size(); i++)
85.     {
86.         List<Integer> edgeList = this.getEdge(i);
87.         if (edgeList.size() != 0)
88.         {
89.             System.out.print(i);
90.             for (int j = 0; j < edgeList.size(); j++)
91.             {
92.                 System.out.print(" -> " + edgeList.get(j));
93.             }
94.             System.out.println();
95.         }
96.     }
97. }
98.}
99.
100. public class CheckDAG
101. {
102.     public static void main(String args[])
103.     {
104.         int v, e, count = 1, to, from;
105.         Scanner sc = new Scanner(System.in);
106.         GraphLinkedList glist;
107.         try
108.         {
109.             System.out.println("Enter the number of vertices: ");
110.             v = sc.nextInt();
111.             System.out.println("Enter the number of edges: ");
112.             e = sc.nextInt();
113.             glist = new GraphLinkedList(v);
114.             System.out.println("Enter the edges in the graph : <from> <to>");
115.             while (count <= e)
116.             {
117.                 to = sc.nextInt();

```

```

118.         from = sc.nextInt();
119.         glist.setEdge(to, from);
120.         count++;
121.     }
122.     glist.printGraph();
123.     System.out
124.         .println("--Processing graph to check whether it is DAG--");
125.     if (glist.checkDAG())
126.     {
127.         System.out
128.             .println("Result: \nGiven graph is DAG (Directed Acyclic Graph).");
129.     }
130.     else
131.     {
132.         System.out
133.             .println("Result: \nGiven graph is not DAG (Directed Acyclic Graph).");
134.     }
135. }
136. catch (Exception E)
137. {
138.     System.out
139.         .println("You are trying to access empty adjacency list of a node.");
140. }
141. sc.close();
142. }
143.

```

Output:

advertisement

```
$ javac CheckDAG.java
$ java CheckDAG
```

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

4 6

5 6

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5 -> 6

5 -> 6

6 -> 3

--Processing graph to check whether it is DAG--

Target Node - 3

Deleting edge between target node 3 - 2

Deleting edge between target node 3 - 6

Target Node - 6

Deleting edge between target node 6 - 4

Deleting edge between target node 6 - 5

Target Node - 5

Deleting edge between target node 5 - 4

Target Node - 4

Deleting edge between target node 4 - 2

Target Node - 2

Deleting edge between target node 2 - 1

Result:

Given graph is DAG (Directed Acyclic Graph).

Enter the number of vertices:

6

Enter the number of edges:

7

Enter the edges in the graph : <from> <to>

1 2

2 3

2 4

4 5

5 6

6 4

6 3

The Graph is:

1 -> 2

2 -> 3 -> 4

4 -> 5

5 -> 6

6 -> 4 -> 3

--Processing graph to check whether it is DAG--

Target Node - 3

Deleting edge between target node 3 - 2

Deleting edge between target node 3 - 6

Result:

Given graph is not DAG (Directed Acyclic Graph).

337.Java Program to Find Minimum Number of Edges to Cut to make the Graph Disconnected

[« Prev](#)

[Next »](#)

This is a java program to find bridges in a graph.

Here is the source code of the Java Program to Find Minimum Number of Edges to Cut to make the Graph Disconnected. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.Iterator;
5. import java.util.NoSuchElementException;
6. import java.util.Scanner;
7. import java.util.Stack;
8.
9. class Bag<Item> implements Iterable<Item>
10. {
11.     private int      N; // number of elements in bag
12.     private Node<Item> first; // beginning of bag
13.
14.     // helper linked list class
15.     private static class Node<Item>
16.     {
17.         private Item      item;
18.         private Node<Item> next;
19.     }
20.
21.     public Bag()
22.     {
23.         first = null;
24.         N = 0;
25.     }
26.
27.     public boolean isEmpty()
28.     {
29.         return first == null;
30.     }
31.
32.     public int size()
33.     {
34.         return N;
35.     }
36.
37.     public void add(Item item)
38.     {
39.         Node<Item> oldfirst = first;
40.         first = new Node<Item>();
41.         first.item = item;
42.         first.next = oldfirst;
43.         N++;
44.     }
45.
46.     public Iterator<Item> iterator()
47.     {
48.         return new ListIterator<Item>(first);
49.     }
50.
51.     // an iterator, doesn't implement remove() since it's optional
52.     @SuppressWarnings("hiding")
```

```

53. private class ListIterator<Item> implements Iterator<Item>
54. {
55.     private Node<Item> current;
56.
57.     public ListIterator(Node<Item> first)
58.     {
59.         current = first;
60.     }
61.
62.     public boolean hasNext()
63.     {
64.         return current != null;
65.     }
66.
67.     public void remove()
68.     {
69.         throw new UnsupportedOperationException();
70.     }
71.
72.     public Item next()
73.     {
74.         if (!hasNext())
75.             throw new NoSuchElementException();
76.         Item item = current.item;
77.         current = current.next;
78.         return item;
79.     }
80. }
81.
82.
83.class BridgeGraph
84. {
85.     private final int V;
86.     private int E;
87.     private Bag<Integer>[] adj;
88.
89.     @SuppressWarnings("unchecked")
90.     public BridgeGraph(int V)
91.     {
92.         if (V < 0)
93.             throw new IllegalArgumentException(
94.                 "Number of vertices must be nonnegative");
95.         this.V = V;
96.         this.E = 0;
97.         adj = (Bag<Integer>[]) new Bag[V];
98.         for (int v = 0; v < V; v++)
99.         {
100.             adj[v] = new Bag<Integer>();
101.         }
102.         System.out.println("Enter the number of edges: ");
103.         Scanner sc = new Scanner(System.in);
104.         int E = sc.nextInt();
105.         if (E < 0)
106.         {
107.             sc.close();
108.             throw new IllegalArgumentException(
109.                 "Number of edges must be nonnegative");
110.         }
111.         for (int i = 0; i < E; i++)
112.         {
113.             int v = sc.nextInt();
114.             int w = sc.nextInt();
115.             addEdge(v, w);
116.         }
117.         sc.close();
118.     }

```

```

119.
120. public BridgeGraph(BridgeGraph G)
121. {
122.     this.V();
123.     this.E = G.E();
124.     for (int v = 0; v < G.V(); v++)
125.     {
126.         // reverse so that adjacency list is in same order as original
127.         Stack<Integer> reverse = new Stack<Integer>();
128.         for (int w : G.adj[v])
129.         {
130.             reverse.push(w);
131.         }
132.         for (int w : reverse)
133.         {
134.             adj[v].add(w);
135.         }
136.     }
137. }
138.
139. public int V()
140. {
141.     return V;
142. }
143.
144. public int E()
145. {
146.     return E;
147. }
148.
149. public void addEdge(int v, int w)
150. {
151.     if (v < 0 || v >= V)
152.         throw new IndexOutOfBoundsException();
153.     if (w < 0 || w >= V)
154.         throw new IndexOutOfBoundsException();
155.     E++;
156.     adj[v].add(w);
157.     adj[w].add(v);
158. }
159.
160. public Iterable<Integer> adj(int v)
161. {
162.     if (v < 0 || v >= V)
163.         throw new IndexOutOfBoundsException();
164.     return adj[v];
165. }
166.
167. public String toString()
168. {
169.     StringBuilder s = new StringBuilder();
170.     String NEWLINE = System.getProperty("line.separator");
171.     s.append(V + " vertices, " + E + " edges " + NEWLINE);
172.     for (int v = 0; v < V; v++)
173.     {
174.         s.append(v + ": ");
175.         for (int w : adj[v])
176.         {
177.             s.append(w + " ");
178.         }
179.         s.append(NEWLINE);
180.     }
181.     return s.toString();
182. }
183. }
184.

```

```

185. public class BridgesinGraph
186. {
187.   private int bridges; // number of bridges
188.   private int cnt; // counter
189.   private int[] pre; // pre[v] = order in which dfs examines v
190.   private int[] low; // low[v] = lowest preorder of any vertex connected
191.           // to v
192.
193.   public BridgesinGraph(BridgeGraph G)
194.   {
195.     low = new int[G.V()];
196.     pre = new int[G.V()];
197.     for (int v = 0; v < G.V(); v++)
198.       low[v] = -1;
199.     for (int v = 0; v < G.V(); v++)
200.       pre[v] = -1;
201.     for (int v = 0; v < G.V(); v++)
202.       if (pre[v] == -1)
203.         dfs(G, v, v);
204.   }
205.
206.   public int components()
207.   {
208.     return bridges + 1;
209.   }
210.
211.   private void dfs(BridgeGraph G, int u, int v)
212.   {
213.     pre[v] = cnt++;
214.     low[v] = pre[v];
215.     for (int w : G.adj(v))
216.     {
217.       if (pre[w] == -1)
218.       {
219.         dfs(G, v, w);
220.         low[v] = Math.min(low[v], low[w]);
221.         if (low[w] == pre[w])
222.         {
223.           System.out.println(v + "-" + w + " is a bridge");
224.           bridges++;
225.         }
226.       }
227.       // update low number - ignore reverse of edge leading to v
228.       else if (w != u)
229.         low[v] = Math.min(low[v], pre[w]);
230.     }
231.   }
232.
233.   public static void main(String[] args)
234.   {
235.     Scanner sc = new Scanner(System.in);
236.     System.out.println("Enter the number of vertices: ");
237.     BridgeGraph G = new BridgeGraph(sc.nextInt());
238.     System.out.println(G);
239.     BridgesinGraph bridge = new BridgesinGraph(G);
240.     System.out
241.       .println("Edge connected components = " + bridge.components());
242.     sc.close();
243.   }
244. }
```

Output:

advertisement

```
$ javac BridgesinGraph.ajav
$ java BridgesinGraph
```

Enter the number of vertices:

6

Enter the number of edges:

7

0 1

1 2

1 3

3 4

4 5

5 3

5 2

6 vertices, 7 edges

0: 1

1: 3 2 0

2: 5 1

3: 5 4 1

4: 5 3

5: 2 3 4

0-1 is a bridge

Edge connected components = 2

338.Java Program to Check if a Given Graph Contain Hamiltonian Cycle or Not

[« Prev](#)

[Next »](#)

This is a java program to check if the graph contains any Hamiltonian cycle. In the mathematical field of graph theory, a Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian path that is a cycle. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem, which is NP-complete.

Here is the source code of the Java Program to Check if a Given Graph Contain Hamiltonian Cycle or Not. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.Arrays;
5. import java.util.Scanner;
6.
7. public class CheckHamiltonianCycle
8. {
9.     private int V, pathCount;
10.    private int[] path;
11.    private int[][] graph;
12.
13.    /** Function to find cycle */
14.    public void findHamiltonianCycle(int[][] g)
15.    {
16.        V = g.length;
17.        path = new int[V];
18.        Arrays.fill(path, -1);
19.        graph = g;
20.        try
21.        {
22.            path[0] = 0;
23.            pathCount = 1;
24.            solve(0);
25.            System.out.println("No solution");
26.        }
27.        catch (Exception e)
28.        {
29.            System.out.println(e.getMessage());
30.            display();
31.        }
32.    }
33.
34.    /** function to find paths recursively */
35.    public void solve(int vertex) throws Exception
36.    {
37.        /** solution */
38.        if (graph[vertex][0] == 1 && pathCount == V)
39.            throw new Exception("Solution found");
40.        /** all vertices selected but last vertex not linked to 0 */
41.        if (pathCount == V)
42.            return;
43.        for (int v = 0; v < V; v++)
44.        {
45.            /** if connected */
46.            if (graph[vertex][v] == 1)
47.            {
48.                /** add to path */
49.                path[pathCount++] = v;
```

```

50.      /** remove connection */
51.      graph[vertex][v] = 0;
52.      graph[v][vertex] = 0;
53.      /** if vertex not already selected solve recursively */
54.      if (!isPresent(v))
55.          solve(v);
56.      /** restore connection */
57.      graph[vertex][v] = 1;
58.      graph[v][vertex] = 1;
59.      /** remove path */
60.      path[--pathCount] = -1;
61.  }
62. }
63. }
64.
65. /** function to check if path is already selected */
66. public boolean isPresent(int v)
67. {
68.     for (int i = 0; i < pathCount - 1; i++)
69.         if (path[i] == v)
70.             return true;
71.     return false;
72. }
73.
74. /** display solution */
75. public void display()
76. {
77.     System.out.print("\nPath : ");
78.     for (int i = 0; i <= V; i++)
79.         System.out.print(path[i % V] + " ");
80.     System.out.println();
81. }
82.
83. /** Main function */
84. public static void main(String[] args)
85. {
86.     Scanner scan = new Scanner(System.in);
87.     /** Make an object of HamiltonianCycle class */
88.     CheckHamiltonianCycle hc = new CheckHamiltonianCycle();
89.     /** Accept number of vertices */
90.     System.out.println("Enter number of vertices");
91.     int V = scan.nextInt();
92.     /** get graph */
93.     System.out.println("Enter adjacency matrix");
94.     int[][] graph = new int[V][V];
95.     for (int i = 0; i < V; i++)
96.         for (int j = 0; j < V; j++)
97.             graph[i][j] = scan.nextInt();
98.     hc.findHamiltonianCycle(graph);
99.     scan.close();
100. }
101.}

```

Output:

advertisement

```
$ javac CheckHamiltonianCycle.java
$ java CheckHamiltonianCycle
```

Enter number of vertices

6

Enter adjacency matrix

```
0 1 0 0 0 0
1 0 1 1 0 0
0 1 0 0 0 1
0 1 0 0 1 1
```

0 0 0 1 0 1

0 0 1 1 1 0

No solution

339.Java Program to Find the Longest Path in a DAG

« [Prev](#)

[Next](#) »

This is a java program to find longest path in DAG.

Here is the source code of the Java Program to Find the Longest Path in a DAG. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. package com.sanfoundry.hardgraph;
2.
3.
4. import java.util.Scanner;
5. import java.util.Vector;
6.
7. class Node
8. {
9.     int name; // node ID, started from 0 to n-1
10.    Vector<Integer> preds; // predecessors (String)
11.    Vector<Integer> neibs; // neighbors (String)
12.    Vector<Integer> backs; // backward edges -node is end vertex (Integer)
13.    Vector<Integer> fors; // forward edges -node is start vertex (Integer)
14.    int pNode; // previous node on the augmenting path
15.    int pEdge; // from which edge this node comes on the augmenting
16.           // path
17.
18.    public Node(int id)
19.    {
20.        name = id;
21.        backs = new Vector<Integer>();
22.        fors = new Vector<Integer>();
23.        pNode = -1;
24.        pEdge = -1;
25.    }
26.
27.
28. class Edge
29. {
30.     int name; // edge ID, started from 0 to n-1
31.     int start; // start vertex of this edge
32.     int end; // end vertex of this edge
33.     int direct; // forwards (+1) or backwards (-1) on augmenting path
34.           // if 0 then not part of augmenting path
35.     int capacity; // capacity
36.     int flow; // current flow
37.
38.     public Edge(int id)
39.     {
40.         name = id;
41.         start = -1;
42.         end = -1;
43.         direct = 0; // default is neither
44.         capacity = 0;
45.         flow = 0;
46.     }
47.
48.     public String toString()
49.     {
50.         return name + ": s=" + start + " e=" + end + " d=" + direct;
51.     }
52.
53.
54. public class LongestPathinDAG
55. {
```

```

56. int n; // number of nodes
57. int target; // destination node
58. int minLength; // the minimal length of each path
59. Node[] v; // used to store Nodes
60. Edge[] e; // used to store Edges
61. int[] path; // used to store temporary path
62. int length = 0; // length of the path
63. int distance = 0; // distance of the path
64. int[] bestPath; // used to store temporary path
65. int bestLength = 0; // length of the longest path
66. int bestDistance = -1000000; // distance of the longest path
67. int[] visited; // used to mark a node as visited if set as
68. // 1
69.
70. public LongestPathinDAG()
71. {
72.     Scanner sc = new Scanner(System.in);
73.     System.out.println("Enter the number of vertices: ");
74.     n = sc.nextInt();
75.     System.out.println("Enter the number of edges: ");
76.     int m = sc.nextInt();
77.     v = new Node[n];
78.     e = new Edge[m];
79.     System.out.println(n + " nodes and " + m + " edges.");
80.     for (int i = 0; i < n; i++)
81.         v[i] = new Node(i);
82.     int i = 0;
83.     while (i < e.length)
84.     {
85.         Edge edge = new Edge(i);
86.         int sVal = sc.nextInt();
87.         edge.start = sVal; // sc.nextInt();
88.         int eVal = sc.nextInt();
89.         edge.end = eVal; // sc.nextInt();
90.         edge.capacity = sc.nextInt();
91.         System.out.println(" edge: " + edge.start + " - " + edge.end
92.             + " : " + edge.capacity);
93.         edge.flow = 0;
94.         e[i] = edge;
95.         v[sVal].fors.add(i);
96.         v[eVal].backs.add(i);
97.         i++;
98.         if (i == m)
99.             break;
100.    }
101.    visited = new int[v.length];
102.    path = new int[v.length];
103.    bestPath = new int[v.length];
104.    sc.close();
105. }
106.
107. /*
108. * this function looks for a longest path starting from begin to end,
109. * using the backtrack depth-first search.
110. */
111. public boolean findLongestPath(int begin, int end, int minLen)
112. {
113.     /*
114.     * compute a longest path from begin to end
115.     */
116.     target = end;
117.     bestDistance = -100000000;
118.     minLength = minLen;
119.     dfsLongestPath(begin);
120.     if (bestDistance == -100000000)
121.         return false;

```

```

122.     else
123.         return true;
124.     }
125.
126.     private void dfsLongestPath(int current)
127.     {
128.         visited[current] = 1;
129.         path[length++] = current;
130.         if (current == target && length >= minLength)
131.         {
132.             if (distance > bestDistance)
133.             {
134.                 for (int i = 0; i < length; i++)
135.                     bestPath[i] = path[i];
136.                 bestLength = length;
137.                 bestDistance = distance;
138.             }
139.         }
140.         else
141.         {
142.             Vector<Integer> fors = v[current].fors;
143.             for (int i = 0; i < fors.size(); i++)
144.             {
145.                 Integer edge_obj = (Integer) fors.elementAt(i);
146.                 int edge = edge_obj.intValue();
147.                 if (visited[e[edge].end] == 0)
148.                 {
149.                     distance += e[edge].capacity;
150.                     dfsLongestPath(e[edge].end);
151.                     distance -= e[edge].capacity;
152.                 }
153.             }
154.         }
155.         visited[current] = 0;
156.         length--;
157.     }
158.
159.     public String toString()
160.     {
161.         String output = "v" + bestPath[0];
162.         for (int i = 1; i < bestLength; i++)
163.             output = output + " -> v" + bestPath[i];
164.         return output;
165.     }
166.
167.     public static void main(String arg[])
168.     {
169.         LongestPathinDAG lp = new LongestPathinDAG();
170.         /*
171.          * find a longest path from vertex 0 to vertex n-1.
172.          */
173.         if (lp.findLongestPath(0, lp.n - 1, 1))
174.             System.out.println("Longest Path is " + lp
175.                               + " and the distance is " + lp.bestDistance);
176.         else
177.             System.out.println("No path from v0 to v" + (lp.n - 1));
178.         /*
179.          * find a longest path from vertex 3 to vertex 5.
180.          */
181.         if (lp.findLongestPath(3, 5, 1))
182.             System.out.println("Longest Path is " + lp
183.                               + " and the distance is " + lp.bestDistance);
184.         else
185.             System.out.println("No path from v3 to v5");
186.         /*
187.          * find a longest path from vertex 5 to vertex 3.

```

```
188.      */
189.      if (lp.findLongestPath(lp.n - 1, 3, 1))
190.          System.out.println("Longest Path is " + lp
191.                  + " and the distance is " + lp.bestDistance);
192.      else
193.          System.out.println("No path from v5 to v3");
194.  }
195.}
```

Output:

advertisement

```
$ javac LongestPathinDAG.java
$ java LongestPathinDAG
```

Enter the number of vertices:

6

Enter the number of edges:

7

6 nodes and 7 edges.

0 1 2

edge: 0 - 1 : 2

1 2 3

edge: 1 - 2 : 3

1 3 4

edge: 1 - 3 : 4

3 4 5

edge: 3 - 4 : 5

4 5 6

edge: 4 - 5 : 6

5 3 7

edge: 5 - 3 : 7

5 2 8

edge: 5 - 2 : 8

Longest Path is v0 -> v1 -> v3 -> v4 -> v5 and the distance is 17

Longest Path is v3 -> v4 -> v5 and the distance is 11

Longest Path is v5 -> v3 and the distance is 7

340.Java Program to Find Hamiltonian Cycle in an UnWeighted Graph

[« Prev](#)

[Next »](#)

This is a java program to find hamilton cycle in graph. In the mathematical field of graph theory, a Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian path that is a cycle. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem, which is NP-complete.

Here is the source code of the Java Program to Find Hamiltonian Cycle in an UnWeighted Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.util.Arrays;
5. import java.util.Scanner;
6.
7. public class HamiltonianCycle
8. {
9.     private int V, pathCount;
10.    private int[] path;
11.    private int[][] graph;
12.
13.    /** Function to find cycle */
14.    public void findHamiltonianCycle(int[][] g)
15.    {
16.        V = g.length;
17.        path = new int[V];
18.        Arrays.fill(path, -1);
19.        graph = g;
20.        try
21.        {
22.            path[0] = 0;
23.            pathCount = 1;
24.            solve(0);
25.            System.out.println("No solution");
26.        }
27.        catch (Exception e)
28.        {
29.            System.out.println(e.getMessage());
30.            display();
31.        }
32.    }
33.
34.    /** function to find paths recursively */
35.    public void solve(int vertex) throws Exception
36.    {
37.        /** solution */
38.        if (graph[vertex][0] == 1 && pathCount == V)
39.            throw new Exception("Solution found");
40.        /** all vertices selected but last vertex not linked to 0 */
41.        if (pathCount == V)
42.            return;
43.        for (int v = 0; v < V; v++)
44.        {
45.            /** if connected */
46.            if (graph[vertex][v] == 1)
47.            {
48.                /** add to path */
49.                path[pathCount++] = v;
```

```

50.     /** remove connection */
51.     graph[vertex][v] = 0;
52.     graph[v][vertex] = 0;
53.     /** if vertex not already selected solve recursively */
54.     if (!isPresent(v))
55.         solve(v);
56.     /** restore connection */
57.     graph[vertex][v] = 1;
58.     graph[v][vertex] = 1;
59.     /** remove path */
60.     path[--pathCount] = -1;
61. }
62. }
63. }
64.
65. /** function to check if path is already selected */
66. public boolean isPresent(int v)
67. {
68.     for (int i = 0; i < pathCount - 1; i++)
69.         if (path[i] == v)
70.             return true;
71.     return false;
72. }
73.
74. /** display solution */
75. public void display()
76. {
77.     System.out.print("\nPath : ");
78.     for (int i = 0; i <= V; i++)
79.         System.out.print(path[i % V] + " ");
80.     System.out.println();
81. }
82.
83. /** Main function */
84. public static void main(String[] args)
85. {
86.     Scanner scan = new Scanner(System.in);
87.     /** Make an object of HamiltonianCycle class */
88.     HamiltonianCycle hc = new HamiltonianCycle();
89.     /** Accept number of vertices */
90.     System.out.println("Enter number of vertices");
91.     int V = scan.nextInt();
92.     /** get graph */
93.     System.out.println("Enter adjacency matrix");
94.     int[][] graph = new int[V][V];
95.     for (int i = 0; i < V; i++)
96.         for (int j = 0; j < V; j++)
97.             graph[i][j] = scan.nextInt();
98.     hc.findHamiltonianCycle(graph);
99.     scan.close();
100. }
101. }

```

Output:

advertisement

```
$ javac HamiltonianCycle.java
$ java HamiltonianCycle
```

Enter number of vertices

6

Enter adjacency matrix

```
0 1 0 0 0 0
1 0 1 1 0 0
0 1 0 0 0 1
0 1 0 0 1 1
```

0 0 0 1 0 1

0 0 1 1 1 0

No solution

341.Java Program to Solve TSP Using Minimum Spanning Trees

[« Prev](#)

[Next »](#)

This is a java program to solve TSP using MST.

Here is the source code of the Java Program to Solve TSP Using Minimum Spanning Trees. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.hardgraph;
3.
4. import java.io.BufferedReader;
5. import java.io.FileReader;
6. import java.io.IOException;
7. import java.util.StringTokenizer;
8.
9. public class TSPUsingMST
10. {
11.     // Arrays to keep track of info. related to each city
12.     private String[] cityName;
13.     private String[] cityState;
14.     private int[] cityLat;
15.     private int[] cityLong;
16.     private int[] cityPop;
17.     // 2-D array to keep track of pairwise distances between cities
18.     private int[][] distances;
19.     // number of cities
20.     private static int numCities;
21.
22.     public TSPUsingMST(int n)
23.     {
24.         numCities = n;
25.         // Allotting the space for each 1-D array
26.         cityName = new String[numCities];
27.         cityState = new String[numCities];
28.         cityLat = new int[numCities];
29.         cityLong = new int[numCities];
30.         cityPop = new int[numCities];
31.         // Allocate space for each 2-D array. These arrays have 0 elements in
32.         // row 0,
33.         // 1 element in row 1, 2 elements in row 2, etc.
34.         distances = new int[numCities][];
35.         for (int i = 0; i < numCities; i++)
36.             distances[i] = new int[i];
37.         try
38.         {
39.             // Construct a buffered reader object and connect it to the files
40.             // "miles.dat"
41.             BufferedReader in = new BufferedReader(new FileReader("miles.dat"));
42.             // A counter that keeps track of the index of the current city being
43.             // read
44.             int cityNumber = 0;
45.             // While-loop for reading in data from "miles.dat." At the beginning
46.             // of the while-loop
47.             // the expectation is that we'll be reading a line containing the
48.             // city name. Instead,
49.             // if we encounter a line that starts with "*" then we skip to the
50.             // next line
51.             while (cityNumber < numCities)
52.             {
53.                 // Read in a line
```

```

54.     String line = in.readLine();
55.     // Skip the rest of the loop if line starts with a "*"
56.     if (line.charAt(0) == '*')
57.         continue;
58.     // Otherwise tokenize the line
59.     StringTokenizer tokenizedLine = new StringTokenizer(line, ",[]");
60.     // Putting actual data into correct position in the array
61.     cityName[cityNumber] = tokenizedLine.nextToken();
62.     cityState[cityNumber] = (tokenizedLine.nextToken()).trim(); // trim()
63.                                         // gets
64.                                         // rid
65.                                         // of
66.                                         // leading/trailing
67.                                         // blanks
68.     cityLat[cityNumber] = Integer.parseInt(tokenizedLine
69.         .nextToken());
70.     cityLong[cityNumber] = Integer.parseInt(tokenizedLine
71.         .nextToken());
72.     cityPop[cityNumber] = Integer.parseInt(tokenizedLine
73.         .nextToken());
74.     // while loop to put distances in the array; this may need to
75.     // read several lines
76.     int mileNumber = 0;
77.     while (mileNumber < cityNumber)
78.     {
79.         // Read a mileage line and tokenize it
80.         String mileage = in.readLine();
81.         StringTokenizer tokenizedMileage = new StringTokenizer(
82.             mileage, " ");
83.         // Read all the mileage data in this line into row
84.         // cityNumber; increment
85.         // mileNumber after each read
86.         while (tokenizedMileage.hasMoreTokens())
87.         {
88.             distances[cityNumber][cityNumber - mileNumber - 1] = Integer
89.                 .parseInt(tokenizedMileage.nextToken());
90.             mileNumber++;
91.         }
92.     } // end of while reading distances
93.     cityNumber++;
94. } // end of while reading cities
95. in.close();
96. } // end of try
97. catch (IOException e)
98. {
99.     System.out.println("File not found.");
100. }
101. } // end of TSPTester() constructor
102.
103. // A simple getIndex method to help test the constructor
104. int getIndex(String city, String state)
105. {
106.     int location;
107.     for (location = 0; location < numCities; location++)
108.         if ((cityName[location].equals(city))
109.             && (cityState[location].equals(state)))
110.             return location;
111.     return -1;
112. }
113.
114. // Print information about a city, given a city index
115. void printCityInfo(int index)
116. {
117.     System.out
118.         .println(cityName[index] + " " + cityState[index] + " "
119.             + cityLat[index] + " " + cityLong[index] + " "

```

```

120.           + cityPop[index]);
121.     }
122.
123. // Print distance information between a given pair of cities
124. void printDistanceInfo(int i, int j)
125. {
126.   if (i < j)
127.     System.out.println(distances[j][i]);
128.   else
129.     System.out.println(distances[i][j]);
130. }
131.
132. int getDistance(int i, int j)
133. {
134.   if (i < j)
135.     return distances[j][i];
136.   else if (j < i)
137.     return distances[i][j];
138.   else
139.     return 0;
140. }
141.
142. int[] greedyTSP()
143. {
144.   // Find a cheapest triangle
145.   // Load triangle 0-1-2 into the first 3 slots of the greedy array
146.   int[] greedy = new int[numCities];
147.   int currentDistance;
148.   greedy[0] = 0;
149.   greedy[1] = 1;
150.   greedy[2] = 2;
151.   int currentBestDistance = getDistance(0, 1) + getDistance(1, 2)
152.       + getDistance(2, 0);
153.   for (int i = 0; i < numCities; i++)
154.     for (int j = 0; j < i; j++)
155.       for (int k = 0; k < j; k++)
156.         if ((currentDistance = getDistance(i, j)
157.             + getDistance(j, k) + getDistance(i, k)) < currentBestDistance)
158.         {
159.           greedy[0] = i;
160.           greedy[1] = j;
161.           greedy[2] = k;
162.           currentBestDistance = currentDistance;
163.         }
164.   // Try greedily to add a city that yields the smallest increase
165.   // in the cost of the tour
166.   int partialTourSize = 3;
167.   boolean[] visited = new boolean[numCities];
168.   for (int i = 0; i < numCities; i++)
169.     visited[i] = false;
170.   visited[greedy[0]] = true;
171.   visited[greedy[1]] = true;
172.   visited[greedy[2]] = true;
173.   // Main loop: keep repeating until partial tour covers all cities
174.   while (partialTourSize < numCities)
175.   {
176.     int smallestIncrease = Integer.MAX_VALUE;
177.     int increase = 0;
178.     int bestInsertionPoint = 0;
179.     int bestCity = 0;
180.     // Scan through all cities, stopping at unvisited cities
181.     for (int i = 0; i < numCities; i++)
182.     {
183.       if (!visited[i])
184.       {
185.         // Consider all possible positions of inserting city i into

```

```

186.          // the tour
187.          // and record the smallest increase
188.          for (int j = 0; j < partialTourSize; j++)
189.          {
190.              increase = getDistance(greedy[j], i)
191.                  + getDistance(i, greedy[(j + 1) % numCities])
192.                  - getDistance(greedy[j], greedy[(j + 1)
193.                                  % numCities]);
194.              if (increase < smallestIncrease)
195.              {
196.                  smallestIncrease = increase;
197.                  bestCity = i;
198.                  bestInsertionPoint = j;
199.              } // end of if we have found a smaller increase
200.          } // end of for-j
201.      } // end of if not visited
202.  } // end of for-i
203.  // Now we are ready to insert the bestCity at the bestInsertionPoint
204.  for (int j = partialTourSize - 1; j > bestInsertionPoint; j--)
205.      greedy[j + 1] = greedy[j];
206.      greedy[bestInsertionPoint + 1] = bestCity;
207.      visited[bestCity] = true;
208.      partialTourSize++;
209.  } // end-while
210.  return greedy;
211. }
212.
213. void copy(int[] source, int[] dest)
214. {
215.     for (int i = 0; i < dest.length; i++)
216.         dest[i] = source[i];
217. }
218.
219. void TSP(int[] R, int partialTourSize, boolean[] visited, int[] T)
220. {
221.     // Base case: we have discovered a tour better than T
222.     if ((partialTourSize == numCities) && (cost(R) < cost(T)))
223.     {
224.         System.out.println("Base case. Tour cost is " + cost(R));
225.         copy(R, T);
226.         return;
227.     }
228.     // Another base case: our partial tour is not worth completing
229.     if (cost(R, partialTourSize) >= cost(T))
230.         return;
231.     // Recursive case: R is not complete and is currently better than T
232.     // and is therefore worth completing
233.     for (int i = 0; i < numCities; i++)
234.     {
235.         if (!visited[i])
236.         {
237.             // System.out.println("Appending " + i);
238.             visited[i] = true;
239.             R[partialTourSize++] = i;
240.             TSP(R, partialTourSize, visited, T);
241.             partialTourSize--;
242.             visited[i] = false;
243.             // System.out.println("Deleting " + i);
244.         }
245.     } // end of for-loop
246. } // end of TSP
247.
248. double cost(int[] tour)
249. {
250.     return cost(tour, tour.length);
251. }

```

```

252.
253.     double cost(int[] tour, int tourSize)
254.     {
255.         double c = 0;
256.         for (int i = 0; i < tourSize - 1; i++)
257.             c = c + getDistance(tour[i], tour[i + 1]);
258.         c = c + getDistance(tour[tourSize - 1], tour[0]);
259.         return c;
260.     }
261.
262. // Main method
263. public static void main(String[] args)
264. {
265.     int n = 15;
266.     TSPUsingMST T = new TSPUsingMST(n);
267.     // Initialize the list of vertices in the tree
268.     // Initially, no one except vertex 0 is in the tree
269.     boolean[] visited = new boolean[n];
270.     for (int i = 0; i < n; i++)
271.         visited[i] = false;
272.     visited[0] = true;
273.     // Initialize the int[] that maintains the tree to default values
274.     // No vertices have parents set, except vertex 0 whose parent is itself
275.     int[] tree = new int[n];
276.     for (int i = 0; i < n; i++)
277.         tree[i] = -1;
278.     tree[0] = 0;
279.     for (int i = 1; i <= n - 1; i++)
280.     {
281.         long minWeight = Long.MAX_VALUE;
282.         int bestVertex = -1;
283.         int bestParent = -1;
284.         for (int j = 0; j < n; j++)
285.         {
286.             for (int k = 0; k < n; k++)
287.             {
288.                 if ((visited[j]) && (!visited[k]))
289.                 {
290.                     if (T.getDistance(j, k) < minWeight)
291.                     {
292.                         minWeight = T.getDistance(j, k);
293.                         bestVertex = k;
294.                         bestParent = j;
295.                     } // end if better distance is found
296.                 } // end if an edge between a visited and an unvisited is
297.                 //found
298.             } // end for-k
299.         } // end for-j
300.         // Update visited and tree
301.         visited[bestVertex] = true;
302.         tree[bestVertex] = bestParent;
303.     } // end for-i
304.     // Printing the MST
305.     for (int i = 1; i < n; i++)
306.         System.out.println(T.cityName[i] + " " + T.cityState[i] + ", "
307.             + T.cityName[tree[i]] + " " + T.cityState[tree[i]]);
308.     // Computing the MST cost
309.     long cost = 0;
310.     for (int i = 0; i < n; i++)
311.         cost += T.getDistance(i, tree[i]);
312.     System.out.println("The cost of the minimum spanning tree is " + cost);
313. } // end main method
314. } // end class

```

Output:

advertisement

```
$ javac TSPUsingMST.java
```

```
$ java TSPUsingMST
```

Yankton SD, Wisconsin Dells WI

Yakima WA, Williston ND

Worcester MA, Wilmington DE

Wisconsin Dells WI, Youngstown OH

Winston-Salem NC, Winchester VA

Winnipeg MB, Yankton SD

Winchester VA, Wilmington DE

Wilmington NC, Winston-Salem NC

Wilmington DE, Williamsport PA

Williston ND, Winnipeg MB

Williamsport PA, Youngstown OH

Williamson WV, Winston-Salem NC

Wichita Falls TX, Wichita KS

Wichita KS, Yankton SD

The cost of the minimum spanning tree is 5461

342.Java Program to Solve Travelling Salesman Problem for Unweighted Graph

[« Prev](#)

[Next »](#)

This is a java program to solve TSP. The travelling salesman problem (TSP) asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Here is the source code of the Java Program to Solve Travelling Salesman Problem for Unweighted Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.hinguapps.graph;
3.
4. import java.util.InputMismatchException;
5. import java.util.Scanner;
6.
7. public class TSP
8. {
9.     private int      numberOfNodes;
10.    private Stack<Integer> stack;
11.
12.    public TSP()
13.    {
14.        stack = new Stack<Integer>();
15.    }
16.
17.    public void tsp(int adjacencyMatrix[][])
18.    {
19.        numberOfNodes = adjacencyMatrix[1].length - 1;
20.        int[] visited = new int[numberOfNodes + 1];
21.        visited[1] = 1;
22.        stack.push(1);
23.        int element, dst = 0, i;
24.        int min = Integer.MAX_VALUE;
25.        boolean minFlag = false;
26.        System.out.print(1 + "\t");
27.        while (!stack.isEmpty())
28.        {
29.            element = stack.peek();
30.            i = 1;
31.            min = Integer.MAX_VALUE;
32.            while (i <= numberOfNodes)
33.            {
34.                if (adjacencyMatrix[element][i] > 1 && visited[i] == 0)
35.                {
36.                    if (min > adjacencyMatrix[element][i])
37.                    {
38.                        min = adjacencyMatrix[element][i];
39.                        dst = i;
40.                        minFlag = true;
41.                    }
42.                }
43.                i++;
44.            }
45.            if (minFlag)
46.            {
47.                visited[dst] = 1;
48.                stack.push(dst);
49.                System.out.print(dst + "\t");
50.            }
51.        }
52.    }
53. }
```

```

50.         minFlag = false;
51.         continue;
52.     }
53.     stack.pop();
54. }
55. }
56.
57. public static void main(String... arg)
58. {
59.     int number_of_nodes;
60.     Scanner scanner = null;
61.     try
62.     {
63.         System.out.println("Enter the number of nodes in the graph");
64.         scanner = new Scanner(System.in);
65.         number_of_nodes = scanner.nextInt();
66.         int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
67.         System.out.println("Enter the adjacency matrix");
68.         for (int i = 1; i <= number_of_nodes; i++)
69.         {
70.             for (int j = 1; j <= number_of_nodes; j++)
71.             {
72.                 adjacency_matrix[i][j] = scanner.nextInt();
73.             }
74.         }
75.         for (int i = 1; i <= number_of_nodes; i++)
76.         {
77.             for (int j = 1; j <= number_of_nodes; j++)
78.             {
79.                 if (adjacency_matrix[i][j] == 1
80.                     && adjacency_matrix[j][i] == 0)
81.                 {
82.                     adjacency_matrix[j][i] = 1;
83.                 }
84.             }
85.         }
86.         System.out.println("The cities are visited as follows: ");
87.         TSP tspNearestNeighbour = new TSP();
88.         tspNearestNeighbour.tsp(adjacency_matrix);
89.     }
90.     catch (InputMismatchException inputMismatch)
91.     {
92.         System.out.println("Wrong Input format");
93.     }
94.     scanner.close();
95. }
96. }

```

Output:

advertisement

```
$ javac TSP.java
$ java TSP
```

```
Enter the number of nodes in the graph
9
Enter the adjacency matrix
000 374 200 223 108 178 252 285 240 356
374 000 255 166 433 199 135 095 136 017
200 255 000 128 277 128 180 160 131 247
223 166 128 000 430 047 052 084 040 155
108 433 277 430 000 453 478 344 389 423
178 199 128 047 453 000 091 110 064 181
252 135 180 052 478 091 000 114 083 117
285 095 160 084 344 110 114 000 047 078
240 136 131 040 389 064 083 047 000 118
```

356 017 247 155 423 181 117 078 118 000

The cities are visited as follows:

1 5 3 2 9 7 4 6 8

343.Java Program to Show the Duality Transformation of Line and Point

[« Prev](#)

[Next »](#)

This is a java program to show the duality transformation of line and point. The transformation corresponds from line to point and point to line.

Here is the source code of the Java Program to Show the Duality Transformation of Line and Point. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.computationalgeometry;
3.
4. import java.util.Scanner;
5.
6. public class DualityTransformationofPointandLine
7. {
8.     public static void performLineTransformation(double a, double b)
9.     {
10.         System.out.println("X: " + (b / a) + ", Y: " + (b * -1));
11.     }
12.
13.     public static void performPointTransformation(double x, double y)
14.     {
15.         System.out.println("y=" + (-1 * y / x) + "x +" + (-1 * y));
16.     }
17.
18.     public static void main(String[] args)
19.     {
20.         System.out
21.             .println("Perform what transformation.\n1. Line Transformation\n2. Point Transformation");
22.         Scanner sc = new Scanner(System.in);
23.         int option = sc.nextInt();
24.         switch (option)
25.         {
26.             case 1:
27.                 System.out.println("Enter the coefficients of line <y=ax-b>");
28.                 double a = sc.nextDouble();
29.                 double b = sc.nextDouble();
30.                 performLineTransformation(a, b);
31.                 break;
32.             case 2:
33.                 System.out.println("Enter the coordinate of point <x, y>");
34.                 double x = sc.nextDouble();
35.                 double y = sc.nextDouble();
36.                 performPointTransformation(x, y);
37.                 break;
38.             default:
39.                 break;
40.         }
41.         sc.close();
42.     }
43. }
```

Output:

advertisement

```
$ javac DualityTransformationofPointandLine.java
$ java DualityTransformationofPointandLine
```

Perform what transformation.

1. Line Transformation

2. Point Transformation

1

Enter the coefficients of line $<y=ax-b>$

1 2

X: 2.0, Y: -2.0

Perform what transformation.

1. Line Transformation

2. Point Transformation

2

Enter the coordinate of point $<x, y>$

2 -2

y=1.0x +2.0

344.Java Program to Find the Longest Subsequence Common to All Sequences in a Set of Sequences

[« Prev](#)

[Next »](#)

This is a java program to implement LCS. The longest common subsequence (LCS) problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). (Note that a subsequence is different from a substring, for the terms of the former need not be consecutive terms of the original sequence.) It is a classic computer science problem, the basis of data comparison programs such as the diff utility, and has applications in bioinformatics.

Here is the source code of the Java Program to Find the Longest Subsequence Common to All Sequences in a Set of Sequences. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.setandstring;
3. 
4. import java.io.BufferedReader;
5. import java.io.InputStreamReader;
6. import java.io.IOException;
7. 
8. public class LongestCommonSubsequencetoSequences
9. {
10.     public static String lcs(String[] strings)
11.     {
12.         if (strings.length == 0)
13.             return "";
14.         if (strings.length == 1)
15.             return strings[0];
16.         int max = -1;
17.         int cacheSize = 1;
18.         for (int i = 0; i < strings.length; i++)
19.         {
20.             cacheSize *= strings[i].length();
21.             if (strings[i].length() > max)
22.                 max = strings[i].length();
23.         }
24.         String[] cache = new String[cacheSize];
25.         int[] indexes = new int[strings.length];
26.         for (int i = 0; i < indexes.length; i++)
27.             indexes[i] = strings[i].length() - 1;
28.         return lcsBack(strings, indexes, cache);
29.     }
30. 
31.     public static String lcsBack(String[] strings, int[] indexes, String[] cache)
32.     {
33.         for (int i = 0; i < indexes.length; i++)
34.             if (indexes[i] == -1)
35.                 return "";
36.         boolean match = true;
37.         for (int i = 1; i < indexes.length; i++)
38.         {
39.             if (strings[0].charAt(indexes[0]) != strings[i].charAt(indexes[i]))
40.             {
41.                 match = false;
42.                 break;
43.             }
44.         }
45.         if (match)
46.         {
47.             int[] newIndexes = new int[indexes.length];
```

```

48.     for (int i = 0; i < indexes.length; i++)
49.         newIndexes[i] = indexes[i] - 1;
50.     String result = lcsBack(strings, newIndexes, cache)
51.         + strings[0].charAt(indexes[0]);
52.     cache[calcCachePos(indexes, strings)] = result;
53.     return result;
54. }
55. else
56. {
57.     String[] subStrings = new String[strings.length];
58.     for (int i = 0; i < strings.length; i++)
59.     {
60.         if (indexes[i] <= 0)
61.             subStrings[i] = "";
62.         else
63.         {
64.             int[] newIndexes = new int[indexes.length];
65.             for (int j = 0; j < indexes.length; j++)
66.                 newIndexes[j] = indexes[j];
67.             newIndexes[i]--;
68.             int cachePos = calcCachePos(newIndexes, strings);
69.             if (cache[cachePos] == null)
70.                 subStrings[i] = lcsBack(strings, newIndexes, cache);
71.             else
72.                 subStrings[i] = cache[cachePos];
73.         }
74.     }
75.     String longestString = "";
76.     int longestlength = 0;
77.     for (int i = 0; i < subStrings.length; i++)
78.     {
79.         if (subStrings[i].length() > longestlength)
80.         {
81.             longestString = subStrings[i];
82.             longestlength = longestString.length();
83.         }
84.     }
85.     cache[calcCachePos(indexes, strings)] = longestString;
86.     return longestString;
87. }
88. }
89.
90. public static int calcCachePos(int[] indexes, String[] strings)
91. {
92.     int factor = 1;
93.     int pos = 0;
94.     for (int i = 0; i < indexes.length; i++)
95.     {
96.         pos += indexes[i] * factor;
97.         factor *= strings[i].length();
98.     }
99.     return pos;
100. }
101.
102. public static void main(String[] args) throws IOException
103. {
104.     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
105.     System.out.println("Longest Common Subsequence Algorithm Test");
106.     System.out.println("Enter string 1");
107.     String str1 = br.readLine();
108.     System.out.println("Enter string 2");
109.     String str2 = br.readLine();
110.     System.out.println("Enter string 3");
111.     String str3 = br.readLine();
112.     System.out.println("Enter string 4");
113.     String str4 = br.readLine();

```

```
114.     String[] str = new String[] { str1, str2, str3, str4 };
115.     System.out.println(lcs(str));
116. }
117. }
```

Output:

advertisement

```
$ javac LongestCommonSubsequencetoSequences.java
$ java LongestCommonSubsequencetoSequences
```

Longest Common Subsequence Algorithm Test

Enter string 1

Longest Common Subsequence

Enter string 2

Common Subsequence

Enter string 3

Shortest Common Subsequence

Enter string 4

Is it Common Subsequence

Common Subsequence

Longest Common Subsequence Algorithm Test

Enter string 1

Sanfoundry

Enter string 2

LearningCenter

Enter string 3

CLinuxJava

Enter string 4

ProgrammingSkills

a

345.Java Program to Implement Affine Cipher

[« Prev](#)

[Next »](#)

This is a java program to implement Affine Cipher. The affine cipher is a type of monoalphabetic substitution cipher, wherein each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which. As such, it has the weaknesses of all substitution ciphers. Each letter is enciphered with the function $(ax+b) \bmod 26$, where b is the magnitude of the shift.

Here is the source code of the Java Program to Implement Affine Cipher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.setandstring;
3. 
4. import java.util.Scanner;
5. 
6. public class AffineCipher
7. {
8.     public static String encryptionMessage(String Msg)
9.     {
10.         String CTxt = "";
11.         int a = 3;
12.         int b = 6;
13.         for (int i = 0; i < Msg.length(); i++)
14.         {
15.             CTxt = CTxt + (char) (((a * Msg.charAt(i)) + b) % 26) + 65);
16.         }
17.         return CTxt;
18.     }
19. 
20.     public static String decryptionMessage(String CTxt)
21.     {
22.         String Msg = "";
23.         int a = 3;
24.         int b = 6;
25.         int a_inv = 0;
26.         int flag = 0;
27.         for (int i = 0; i < 26; i++)
28.         {
29.             flag = (a * i) % 26;
30.             if (flag == 1)
31.             {
32.                 a_inv = i;
33.                 System.out.println(i);
34.             }
35.         }
36.         for (int i = 0; i < CTxt.length(); i++)
37.         {
38.             Msg = Msg + (char) (((a_inv * ((CTxt.charAt(i) - b)) % 26)) + 65);
39.         }
40.         return Msg;
41.     }
42. 
43.     public static void main(String[] args)
44.     {
45.         Scanner sc = new Scanner(System.in);
46.         System.out.println("Enter the message: ");
47.         String message = sc.next();
48.         System.out.println("Message is :" + message);
49.         System.out.println("Encrypted Message is : "
```

```
50.      + encryptionMessage(message));
51.  System.out.println("Decrypted Message is: "
52.      + decryptionMessage(encryptionMessage(message))));
53.  sc.close();
54. }
55.}
```

Output:

advertisement

```
$ javac AffineCipher.java
$ java AffineCipher
```

```
Enter the message:
SANFOUNDRY
Message is :SANFOUNDRY
Encrypted Message is : VTGIJBGCSN
9
Decrypted Message is: SANFOUNDRY
```

346.Java Program to Perform Cryptography Using Transposition Technique

[« Prev](#)

[Next »](#)

This is a java program to implement transposition technique. In cryptography, a transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt.

Here is the source code of the Java Program to Perform Cryptography Using Transposition Technique. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. public class TranspositionCipher
5. {
6.     public static String selectedKey;
7.     public static char sortedKey[];
8.     public static int sortedKeyPos[];
9.
10.    // default constructor define the default key
11.    public TranspositionCipher()
12.    {
13.        selectedKey = "megabuck";
14.        sortedKeyPos = new int[selectedKey.length()];
15.        sortedKey = selectedKey.toCharArray();
16.    }
17.
18.    // Parameterized constructor define the custom key
19.    public TranspositionCipher(String myKey)
20.    {
21.        selectedKey = myKey;
22.        sortedKeyPos = new int[selectedKey.length()];
23.        sortedKey = selectedKey.toCharArray();
24.    }
25.
26.    // To reorder data do the sorting on selected key
27.    public static void doProcessOnKey()
28.    {
29.        // Find position of each character in selected key and arrange it on
30.        // alphabetical order
31.        int min, i, j;
32.        char orginalKey[] = selectedKey.toCharArray();
33.        char temp;
34.        // First Sort the array of selected key
35.        for (i = 0; i < selectedKey.length(); i++)
36.        {
37.            min = i;
38.            for (j = i; j < selectedKey.length(); j++)
39.            {
40.                if (sortedKey[min] > sortedKey[j])
41.                {
42.                    min = j;
43.                }
44.            }
45.            if (min != i)
46.            {
47.                temp = sortedKey[i];
48.                sortedKey[i] = sortedKey[min];
```

```

49.         sortedKey[min] = temp;
50.     }
51. }
52. //Fill the position of array according to alphabetical order
53. for (i = 0; i < selectedKey.length(); i++)
54. {
55.     for (j = 0; j < selectedKey.length(); j++)
56.     {
57.         if (orginalKey[i] == sortedKey[j])
58.             sortedKeyPos[i] = j;
59.     }
60. }
61. }
62.
63. // to encrypt the targeted string
64. public static String doEncryption(String plainText)
65. {
66.     int min, i, j;
67.     char orginalKey[] = selectedKey.toCharArray();
68.     char temp;
69.     doProcessOnKey();
70.     // Generate encrypted message by doing encryption using Transposition
71.     // Cipher
72.     int row = plainText.length() / selectedKey.length();
73.     int extrabit = plainText.length() % selectedKey.length();
74.     int exrow = (extrabit == 0) ? 0 : 1;
75.     int rowtemp = -1, coltemp = -1;
76.     int totallen = (row + exrow) * selectedKey.length();
77.     char pmat[][] = new char[(row + exrow)][(selectedKey.length())];
78.     char encry[] = new char[totallen];
79.     int tempcnt = -1;
80.     row = 0;
81.     for (i = 0; i < totallen; i++)
82.     {
83.         coltemp++;
84.         if (i < plainText.length())
85.         {
86.             if (coltemp == (selectedKey.length()))
87.             {
88.                 row++;
89.                 coltemp = 0;
90.             }
91.             pmat[row][coltemp] = plainText.charAt(i);
92.         }
93.         else
94.         { // do the padding ...
95.             pmat[row][coltemp] = '*';
96.         }
97.     }
98.     int len = -1, k;
99.     for (i = 0; i < selectedKey.length(); i++)
100.    {
101.        for (k = 0; k < selectedKey.length(); k++)
102.        {
103.            if (i == sortedKeyPos[k])
104.            {
105.                break;
106.            }
107.        }
108.        for (j = 0; j <= row; j++)
109.        {
110.            len++;
111.            encry[len] = pmat[j][k];
112.        }
113.    }
114.    String p1 = new String(encry);

```

```

115.     return (new String(p1));
116. }
117.
118. // to decrypt the targeted string
119. public static String doDecryption(String s)
120. {
121.     int min, i, j, k;
122.     char key[] = selectedKey.toCharArray();
123.     char encry[] = s.toCharArray();
124.     char temp;
125.     doProcessOnKey();
126.     // Now generating plain message
127.     int row = s.length() / selectedKey.length();
128.     char pmat[][] = new char[row][(selectedKey.length())];
129.     int tempcnt = -1;
130.     for (i = 0; i < selectedKey.length(); i++)
131.     {
132.         for (k = 0; k < selectedKey.length(); k++)
133.         {
134.             if (i == sortedKeyPos[k])
135.             {
136.                 break;
137.             }
138.         }
139.         for (j = 0; j < row; j++)
140.         {
141.             tempcnt++;
142.             pmat[j][k] = encry[tempcnt];
143.         }
144.     }
145.     // store matrix character in to a single string
146.     char p1[] = new char[row * selectedKey.length()];
147.     k = 0;
148.     for (i = 0; i < row; i++)
149.     {
150.         for (j = 0; j < selectedKey.length(); j++)
151.         {
152.             if (pmat[i][j] != '*')
153.             {
154.                 p1[k++] = pmat[i][j];
155.             }
156.         }
157.     }
158.     p1[k++] = '\0';
159.     return (new String(p1));
160. }
161.
162. @SuppressWarnings("static-access")
163. public static void main(String[] args)
164. {
165.     TranspositionCipher tc = new TranspositionCipher();
166.     System.out.println("Encrypted Message is: "
167.                         + tc.doEncryption("Sanfoundry"));
168.     System.out.println("Decrypted Message is: "
169.                         + tc.doDecryption(tc.doEncryption("Sanfoundry")));
170. }
171. }
```

Output:

advertisement

```
$ javac TranspositionCipher.java
$ java TranspositionCipher
```

```
Encrypted Message is: f*n*o*a*y*d*Sru*
Decrypted Message is: Sanfoundry
```

347.Java Program to Implement the Vigenere Cypher

[« Prev](#)

[Next »](#)

This is a java program to implement Vigenere cipher. The Vigenère cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword. It is a simple form of polyalphabetic substitution.

Here is the source code of the Java Program to Implement the Vigenere Cypher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.setandstring;
3. 
4. public class VigenereCipher
5. {
6.     public static String encrypt(String text, final String key)
7.     {
8.         String res = "";
9.         text = text.toUpperCase();
10.        for (int i = 0, j = 0; i < text.length(); i++)
11.        {
12.            char c = text.charAt(i);
13.            if (c < 'A' || c > 'Z')
14.                continue;
15.            res += (char) ((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
16.            j = ++j % key.length();
17.        }
18.        return res;
19.    }
20. 
21.    public static String decrypt(String text, final String key)
22.    {
23.        String res = "";
24.        text = text.toUpperCase();
25.        for (int i = 0, j = 0; i < text.length(); i++)
26.        {
27.            char c = text.charAt(i);
28.            if (c < 'A' || c > 'Z')
29.                continue;
30.            res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');
31.            j = ++j % key.length();
32.        }
33.        return res;
34.    }
35. 
36.    public static void main(String[] args)
37.    {
38.        String key = "VIGENRECIPHER";
39.        String message = "Beware the Jabberwock, my son! The jaws that bite, the claws that catch!";
40.        String encryptedMsg = encrypt(message, key);
41.        System.out.println("String: " + message);
42.        System.out.println("Encrypted message: " + encryptedMsg);
43.        System.out.println("Decrypted message: " + decrypt(encryptedMsg, key));
44.    }
45. }
```

Output:

advertisement

```
$ javac VigenereCipher.java
$ java VigenereCipher
```

String: Beware the Jabberwock, my son! The jaws that bite, the claws that catch!

Encrypted message: WMCEEIKLGRPIFVMEUGXQPWQVIOIAVEYXUEKFBTALVXTGAFXYEVKPAGY

Decrypted message: BEWARETHEJABBERWOCKMYSONTHEJAWSTHATBITETHECLAWSTHATCATCH

348.Java Program to Implement the Hill Cypher

« Prev

Next »

This is a java program to implement hill cipher. In classical cryptography, the Hill cipher is a polygraphic substitution cipher based on linear algebra. Invented by Lester S. Hill in 1929, it was the first polygraphic cipher in which it was practical (though barely) to operate on more than three symbols at once. The following discussion assumes an elementary knowledge of matrices.

Here is the source code of the Java Program to Implement the Hill Cypher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.io.BufferedReader;
5. import java.io.IOException;
6. import java.io.InputStreamReader;
7.
8. public class HillCipher
9. {
10.     int keymatrix[][];
11.     int linematrix[];
12.     int resultmatrix[];
13.
14.     public void divide(String temp, int s)
15.     {
16.         while (temp.length() > s)
17.         {
18.             String sub = temp.substring(0, s);
19.             temp = temp.substring(s, temp.length());
20.             perform(sub);
21.         }
22.         if (temp.length() == s)
23.             perform(temp);
24.         else if (temp.length() < s)
25.         {
26.             for (int i = temp.length(); i < s; i++)
27.                 temp = temp + 'x';
28.             perform(temp);
29.         }
30.     }
31.
32.     public void perform(String line)
33.     {
34.         linetomatrix(line);
35.         linemultiplykey(line.length());
36.         result(line.length());
37.     }
38.
39.     public void keytomatrix(String key, int len)
40.     {
41.         keymatrix = new int[len][len];
42.         int c = 0;
43.         for (int i = 0; i < len; i++)
44.         {
45.             for (int j = 0; j < len; j++)
46.             {
47.                 keymatrix[i][j] = ((int) key.charAt(c)) - 97;
48.                 c++;
49.             }
50.         }
51.     }
```

```

52.
53. public void linetomatrix(String line)
54. {
55.     linematrix = new int[line.length()];
56.     for (int i = 0; i < line.length(); i++)
57.     {
58.         linematrix[i] = ((int) line.charAt(i)) - 97;
59.     }
60. }
61.
62. public void linemultiplykey(int len)
63. {
64.     resultmatrix = new int[len];
65.     for (int i = 0; i < len; i++)
66.     {
67.         for (int j = 0; j < len; j++)
68.         {
69.             resultmatrix[i] += keymatrix[i][j] * linematrix[j];
70.         }
71.         resultmatrix[i] %= 26;
72.     }
73. }
74.
75. public void result(int len)
76. {
77.     String result = "";
78.     for (int i = 0; i < len; i++)
79.     {
80.         result += (char) (resultmatrix[i] + 97);
81.     }
82.     System.out.print(result);
83. }
84.
85. public boolean check(String key, int len)
86. {
87.     keytomatrix(key, len);
88.     int d = determinant(keymatrix, len);
89.     d = d % 26;
90.     if (d == 0)
91.     {
92.         System.out
93.             .println("Invalid key!!! Key is not invertible because determinant=0...");
94.         return false;
95.     }
96.     else if (d % 2 == 0 || d % 13 == 0)
97.     {
98.         System.out
99.             .println("Invalid key!!! Key is not invertible because determinant has common factor with 26...");
100.        return false;
101.    }
102.    else
103.    {
104.        return true;
105.    }
106. }
107.
108. public int determinant(int A[][], int N)
109. {
110.     int res;
111.     if (N == 1)
112.         res = A[0][0];
113.     else if (N == 2)
114.     {
115.         res = A[0][0] * A[1][1] - A[1][0] * A[0][1];
116.     }
117.     else

```

```

118.    {
119.        res = 0;
120.        for (int j1 = 0; j1 < N; j1++)
121.        {
122.            int m[][] = new int[N - 1][N - 1];
123.            for (int i = 1; i < N; i++)
124.            {
125.                int j2 = 0;
126.                for (int j = 0; j < N; j++)
127.                {
128.                    if (j == j1)
129.                        continue;
130.                    m[i - 1][j2] = A[i][j];
131.                    j2++;
132.                }
133.            }
134.            res += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
135.                * determinant(m, N - 1);
136.        }
137.    }
138.    return res;
139. }
140.
141. public void cofact(int num[][], int f)
142. {
143.     int b[][], fac[][];
144.     b = new int[f][f];
145.     fac = new int[f][f];
146.     int p, q, m, n, i, j;
147.     for (q = 0; q < f; q++)
148.     {
149.         for (p = 0; p < f; p++)
150.         {
151.             m = 0;
152.             n = 0;
153.             for (i = 0; i < f; i++)
154.             {
155.                 for (j = 0; j < f; j++)
156.                 {
157.                     b[i][j] = 0;
158.                     if (i != q && j != p)
159.                     {
160.                         b[m][n] = num[i][j];
161.                         if (n < (f - 2))
162.                             n++;
163.                         else
164.                         {
165.                             n = 0;
166.                             m++;
167.                         }
168.                     }
169.                 }
170.             }
171.             fac[q][p] = (int) Math.pow(-1, q + p) * determinant(b, f - 1);
172.         }
173.     }
174.     trans(fac, f);
175. }
176.
177. void trans(int fac[][], int r)
178. {
179.     int i, j;
180.     int b[][], inv[][];
181.     b = new int[r][r];
182.     inv = new int[r][r];
183.     int d = determinant(keymatrix, r);

```

```

184.     int mi = mi(d % 26);
185.     mi %= 26;
186.     if (mi < 0)
187.         mi += 26;
188.     for (i = 0; i < r; i++)
189.     {
190.         for (j = 0; j < r; j++)
191.         {
192.             b[i][j] = fac[j][i];
193.         }
194.     }
195.     for (i = 0; i < r; i++)
196.     {
197.         for (j = 0; j < r; j++)
198.         {
199.             inv[i][j] = b[i][j] % 26;
200.             if (inv[i][j] < 0)
201.                 inv[i][j] += 26;
202.             inv[i][j] *= mi;
203.             inv[i][j] %= 26;
204.         }
205.     }
206.     System.out.println("\nInverse key:");
207.     matrixtoinvkey(inv, r);
208. }
209.
210. public int mi(int d)
211. {
212.     int q, r1, r2, r, t1, t2, t;
213.     r1 = 26;
214.     r2 = d;
215.     t1 = 0;
216.     t2 = 1;
217.     while (r1 != 1 && r2 != 0)
218.     {
219.         q = r1 / r2;
220.         r = r1 % r2;
221.         t = t1 - (t2 * q);
222.         r1 = r2;
223.         r2 = r;
224.         t1 = t2;
225.         t2 = t;
226.     }
227.     return (t1 + t2);
228. }
229.
230. public void matrixtoinvkey(int inv[][], int n)
231. {
232.     String invkey = "";
233.     for (int i = 0; i < n; i++)
234.     {
235.         for (int j = 0; j < n; j++)
236.         {
237.             invkey += (char) (inv[i][j] + 97);
238.         }
239.     }
240.     System.out.print(invkey);
241. }
242.
243. public static void main(String args[]) throws IOException
244. {
245.     HillCipher obj = new HillCipher();
246.     BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
247.     int choice;
248.     System.out.println("Menu:\n1: Encryption\n2: Decryption");
249.     choice = Integer.parseInt(in.readLine());

```

```
250.     System.out.println("Enter the line: ");
251.     String line = in.readLine();
252.     System.out.println("Enter the key: ");
253.     String key = in.readLine();
254.     double sq = Math.sqrt(key.length());
255.     if (sq != (long) sq)
256.         System.out
257.             .println("Invalid key length!!! Does not form a square matrix...");  
258.     else
259.     {
260.         int s = (int) sq;
261.         if (obj.check(key, s))
262.         {
263.             System.out.println("Result:");
264.             obj.divide(line, s);
265.             obj.cofact(obj.keymatrix, s);
266.         }
267.     }
268. }
269. }
```

Output:

advertisement

```
$ javac HillCipher.java
$ java HillCipher
```

Menu:

1: Encryption
2: Decryption

1

Enter the line:

sanfoundry

Enter the key:

sanfoundr

Result:

zmmnmxfnfzdss

Inverse key:

inabzfjeq

Menu:

1: Encryption

2: Decryption

2

Enter the line:

zmmnmxfnfzdss

Enter the key:

inabzfjeq

Result:

sanfoundry

Inverse key:

sanfoundr

349.Java Program to Implement the Monoalphabetic Cypher

[« Prev](#)

[Next »](#)

This is a java program to implement monoalphabetic cypher. In cryptography, a substitution cipher is a method of encoding by which units of plaintext are replaced with ciphertext, according to a regular system; the “units” may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.

Here is the source code of the Java Program to Implement the Monoalphabetic Cypher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class MonoalphabeticCipher
7. {
8.     public static char p[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
9.         'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
10.        'w', 'x', 'y', 'z' };
11.    public static char ch[] = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'T', 'O',
12.        'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z', 'X', 'C',
13.        'V', 'B', 'N', 'M' };
14.
15.    public static String doEncryption(String s)
16.    {
17.        char c[] = new char[(s.length())];
18.        for (int i = 0; i < s.length(); i++)
19.        {
20.            for (int j = 0; j < 26; j++)
21.            {
22.                if (p[j] == s.charAt(i))
23.                {
24.                    c[i] = ch[j];
25.                    break;
26.                }
27.            }
28.        }
29.        return (new String(c));
30.    }
31.
32.    public static String doDecryption(String s)
33.    {
34.        char p1[] = new char[(s.length())];
35.        for (int i = 0; i < s.length(); i++)
```

```
36.     {
37.         for (int j = 0; j < 26; j++)
38.         {
39.             if (ch[j] == s.charAt(i))
40.             {
41.                 p1[i] = p[j];
42.                 break;
43.             }
44.         }
45.     }
46.     return (new String(p1));
47. }
48.
49. public static void main(String args[])
50. {
51.     Scanner sc = new Scanner(System.in);
52.     System.out.println("Enter the message: ");
53.     String en = doEncryption(sc.nextLine().toLowerCase());
54.     System.out.println("Encrypted message: " + en);
55.     System.out.println("Decrypted message: " + doDecryption(en));
56.     sc.close();
57. }
58.}
```

Output:

```
$ javac MonoalphabeticCipher.java
$ java MonoalphabeticCipher
```

```
Enter the message:
Sanfoundry
Encrypted message: LQFYGXFRKN
Decrypted message: sanfoundry
```

350.Java Program to Implement the Checksum Method for Small String Messages and Detect If the Received message is same as the Transmitted

[« Prev](#)

[Next »](#)

This is a java program to implement checksum method and verify message. A checksum or hash sum is a small-size datum from an arbitrary block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves checksums are often used to verify data integrity, but should not be relied upon to also verify data authenticity.

The actual procedure which yields the checksum, given a data input is called a checksum function or checksum algorithm. Depending on its design goals, a good checksum algorithm will usually output a significantly different value, even for small changes made to the input. This is especially true of cryptographic hash functions, which may be used to detect many data corruption errors and verify overall data integrity; if the computed checksum for the current data input matches the stored value of a previously computed checksum, there is a very high probability the data has not been accidentally altered or corrupted.

Here is the source code of the Java Program to Implement the Checksum Method for Small String Messages and Detect If the Received message is same as the Transmitted. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

advertisement

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.*;
5.
6. public class ChecksumMethod
7. {
8.     public static void main(String args[])
9.     {
10.         Scanner scan = new Scanner(System.in);
11.         System.out.println("Enter the string input:");
12.         String input = scan.nextLine();
13.         int checksum = generateChecksum(input);
14.         // Call the method to create the checksum
15.         System.out.println("The checksum generated is = "
16.             + Integer.toHexString(checksum));
17.         System.out.println("Enter the data to be sent:");
18.         input = scan.nextLine();
19.         System.out.println("Enter the checksum to be sent:");
20.         checksum = Integer.parseInt(scan.nextLine(), 16);
21.         // User inputs data as hexadecimal value but it will be stored as a
22.         // decimal value unless it is converted into hexadecimal first.
23.         receive(input, checksum);
24.         scan.close();
25.     }
26.
27.     static int generateChecksum(String s)
28.     {
29.         String hex_value = new String();
30.         // 'hex_value' will be used to store various hex values as a string
31.         int x, i, checksum = 0;
32.         // 'x' will be used for general purpose storage of integer values
33.         // 'i' is used for loops
34.         // 'checksum' will store the final checksum
```

```

35.     for (i = 0; i < s.length() - 2; i = i + 2)
36.     {
37.         x = (int) (s.charAt(i));
38.         hex_value = Integer.toHexString(x);
39.         x = (int) (s.charAt(i + 1));
40.         hex_value = hex_value + Integer.toHexString(x);
41.         // Extract two characters and get their hexadecimal ASCII values
42.         System.out.println(s.charAt(i) + "" + s.charAt(i + 1) + ":" +
43.             + hex_value);
44.         x = Integer.parseInt(hex_value, 16);
45.         // Convert the hex_value into int and store it
46.         checksum += x;
47.         // Add 'x' into 'checksum'
48.     }
49.     if (s.length() % 2 == 0)
50.     {
51.         // If number of characters is even, then repeat above loop's steps
52.         // one more time.
53.         x = (int) (s.charAt(i));
54.         hex_value = Integer.toHexString(x);
55.         x = (int) (s.charAt(i + 1));
56.         hex_value = hex_value + Integer.toHexString(x);
57.         System.out.println(s.charAt(i) + "" + s.charAt(i + 1) + ":" +
58.             + hex_value);
59.         x = Integer.parseInt(hex_value, 16);
60.     }
61.     else
62.     {
63.         // If number of characters is odd, last 2 digits will be 00.
64.         x = (int) (s.charAt(i));
65.         hex_value = "00" + Integer.toHexString(x);
66.         x = Integer.parseInt(hex_value, 16);
67.         System.out.println(s.charAt(i) + ":" + hex_value);
68.     }
69.     checksum += x;
70.     // Add the generated value of 'x' from the if-else case into 'checksum'
71.     hex_value = Integer.toHexString(checksum);
72.     // Convert into hexadecimal string
73.     if (hex_value.length() > 4)
74.     {
75.         // If a carry is generated, then we wrap the carry
76.         int carry = Integer.parseInt("'" + hex_value.charAt(0)), 16);
77.         // Get the value of the carry bit
78.         hex_value = hex_value.substring(1, 5);
79.         // Remove it from the string
80.         checksum = Integer.parseInt(hex_value, 16);
81.         // Convert it into an int
82.         checksum += carry;
83.         // Add it to the checksum
84.     }
85.     checksum = generateComplement(checksum);
86.     // Get the complement
87.     return checksum;
88. }
89.
90. static void receive(String s, int checksum)
91. {
92.     int generated_checksum = generateChecksum(s);
93.     // Calculate checksum of received data
94.     generated_checksum = generateComplement(generated_checksum);
95.     // Then get its complement, since generated checksum is complemented
96.     int syndrome = generated_checksum + checksum;
97.     // Syndrome is addition of the 2 checksums
98.     syndrome = generateComplement(syndrome);
99.     // It is complemented
100.    System.out.println("Syndrome = " + Integer.toHexString(syndrome));

```

```
101.     if (syndrome == 0)
102.     {
103.         System.out.println("Data is received without error.");
104.     }
105.     else
106.     {
107.         System.out.println("There is an error in the received data.");
108.     }
109. }
110.
111. static int generateComplement(int checksum)
112. {
113.     // Generates 15's complement of a hexadecimal value
114.     checksum = Integer.parseInt("FFFF", 16) - checksum;
115.     return checksum;
116. }
117. }
```

Output:

```
$ javac ChecksumMethod.java
$ java ChecksumMethod
```

Enter the string input:

Sanfoundry

Sa : 5361

nf : 6e66

ou : 6f75

nd : 6e64

ry : 7279

The checksum generated is = ede4

Enter the data to be sent:

Sanfoundry

Enter the checksum to be sent:

ede4

Sa : 5361

nf : 6e66

ou : 6f75

nd : 6e64

ry : 7279

Syndrome = 0

Data is received without error.

Enter the string input:

Sanfoundry

Sa : 5361

nf : 6e66

ou : 6f75

nd : 6e64

ry : 7279

The checksum generated is = ede4

Enter the data to be sent:

sanfoundry

Enter the checksum to be sent:

ede4

sa : 7361

nf : 6e66

ou : 6f75

nd : 6e64

ry : 7279

Syndrome = fffffe000

There is an error in the received data.

351.Java Program to Implement the RSA Algorithm

[« Prev](#)

[Next »](#)

This is a java program to implement RSA algorithm. RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman.

Here is the source code of the Java Program to Implement the RSA Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.io.DataInputStream;
5. import java.io.IOException;
6. import java.math.BigInteger;
7. import java.util.Random;
8.
9. public class RSA
10. {
11.     private BigInteger p;
12.     private BigInteger q;
13.     private BigInteger N;
14.     private BigInteger phi;
15.     private BigInteger e;
16.     private BigInteger d;
17.     private int bitlength = 1024;
18.     private Random r;
19.
20.     public RSA()
21.     {
22.         r = new Random();
23.         p = BigInteger.probablePrime(bitlength, r);
24.         q = BigInteger.probablePrime(bitlength, r);
25.         N = p.multiply(q);
26.         phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
27.         e = BigInteger.probablePrime(bitlength / 2, r);
28.         while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
29.         {
30.             e.add(BigInteger.ONE);
31.         }
32.         d = e.modInverse(phi);
33.     }
34.
35.     public RSA(BigInteger e, BigInteger d, BigInteger N)
36.     {
37.         this.e = e;
38.         this.d = d;
39.         this.N = N;
40.     }
41.
42.     @SuppressWarnings("deprecation")
43.     public static void main(String[] args) throws IOException
44.     {
45.         RSA rsa = new RSA();
46.         DataInputStream in = new DataInputStream(System.in);
47.         String teststring;
48.         System.out.println("Enter the plain text:");
49.         teststring = in.readLine();
50.         System.out.println("Encrypting String: " + teststring);
```

```

51. System.out.println("String in Bytes: "
52.         + bytesToString(teststring.getBytes()));
53. // encrypt
54. byte[] encrypted = rsa.encrypt(teststring.getBytes());
55. // decrypt
56. byte[] decrypted = rsa.decrypt(encrypted);
57. System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
58. System.out.println("Decrypted String: " + new String(decrypted));
59. }
60.
61. private static String bytesToString(byte[] encrypted)
62. {
63.     String test = "";
64.     for (byte b : encrypted)
65.     {
66.         test += Byte.toString(b);
67.     }
68.     return test;
69. }
70.
71. // Encrypt message
72. public byte[] encrypt(byte[] message)
73. {
74.     return (new BigInteger(message)).modPow(e, N).toByteArray();
75. }
76.
77. // Decrypt message
78. public byte[] decrypt(byte[] message)
79. {
80.     return (new BigInteger(message)).modPow(d, N).toByteArray();
81. }
82.

```

Output:

advertisement

```
$ javac RSA.java
$ java RSA
```

Enter the plain text:

```
Sanfoundry
Encrypting String: Sanfoundry
String in Bytes: 8397110102111117110100114121
Decrypting Bytes: 8397110102111117110100114121
Decrypted String: Sanfoundry
```

352.Java Program to Implement the MD5 Algorithm

[« Prev](#)

[Next »](#)

This is a java program to implement MD5 algorithm. The MD5 message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity.

Here is the source code of the Java Program to Implement the MD5 Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. public class MD5
5. {
6.     private static final int INIT_A = 0x67452301;
7.     private static final int INIT_B = (int) 0xEFCDAB89L;
8.     private static final int INIT_C = (int) 0x98BADC6E;
9.     private static final int INIT_D = 0x10325476;
10.    private static final int[] SHIFT_AMTS = { 7, 12, 17, 22, 5, 9, 14, 20, 4,
11.        11, 16, 23, 6, 10, 15, 21 };
12.    private static final int[] TABLE_T = new int[64];
13.    static
14.    {
15.        for (int i = 0; i < 64; i++)
16.            TABLE_T[i] = (int) (long) ((1L << 32) * Math.abs(Math.sin(i + 1)));
17.    }
18.
19.    public static byte[] computeMD5(byte[] message)
20.    {
21.        int messageLenBytes = message.length;
22.        int numBlocks = ((messageLenBytes + 8) >>> 6) + 1;
23.        int totalLen = numBlocks << 6;
24.        byte[] paddingBytes = new byte[totalLen - messageLenBytes];
25.        paddingBytes[0] = (byte) 0x80;
26.        long messageLenBits = (long) messageLenBytes << 3;
27.        for (int i = 0; i < 8; i++)
28.        {
29.            paddingBytes[paddingBytes.length - 8 + i] = (byte) messageLenBits;
30.            messageLenBits >>>= 8;
31.        }
32.        int a = INIT_A;
33.        int b = INIT_B;
34.        int c = INIT_C;
35.        int d = INIT_D;
36.        int[] buffer = new int[16];
37.        for (int i = 0; i < numBlocks; i++)
38.        {
39.            int index = i << 6;
40.            for (int j = 0; j < 64; j++, index++)
41.                buffer[j >>> 2] = (int) ((index < messageLenBytes) ? message[index]
42.                    : paddingBytes[index - messageLenBytes] << 24)
43.                    | (buffer[j >>> 2] >>> 8);
44.            int originalA = a;
45.            int originalB = b;
46.            int originalC = c;
47.            int originalD = d;
48.            for (int j = 0; j < 64; j++)
49.            {
50.                int div16 = j >>> 4;
51.                int f = 0;
```

```

52.     int bufferIndex = j;
53.     switch (div16)
54.     {
55.         case 0:
56.             f = (b & c) | (~b & d);
57.             break;
58.         case 1:
59.             f = (b & d) | (c & ~d);
60.             bufferIndex = (bufferIndex * 5 + 1) & 0x0F;
61.             break;
62.         case 2:
63.             f = b ^ c ^ d;
64.             bufferIndex = (bufferIndex * 3 + 5) & 0x0F;
65.             break;
66.         case 3:
67.             f = c ^ (b | ~d);
68.             bufferIndex = (bufferIndex * 7) & 0x0F;
69.             break;
70.     }
71.     int temp = b
72.         + Integer.rotateLeft(a + f + buffer[bufferIndex]
73.             + TABLE_T[j],
74.             SHIFT_AMTS[(div16 << 2) | (j & 3)]);
75.     a = d;
76.     d = c;
77.     c = b;
78.     b = temp;
79. }
80. a += originalA;
81. b += originalB;
82. c += originalC;
83. d += originalD;
84. }
85. byte[] md5 = new byte[16];
86. int count = 0;
87. for (int i = 0; i < 4; i++)
88. {
89.     int n = (i == 0) ? a : ((i == 1) ? b : ((i == 2) ? c : d));
90.     for (int j = 0; j < 4; j++)
91.     {
92.         md5[count++] = (byte) n;
93.         n >>>= 8;
94.     }
95. }
96. return md5;
97. }
98.
99. public static String toHexString(byte[] b)
100. {
101.     StringBuilder sb = new StringBuilder();
102.     for (int i = 0; i < b.length; i++)
103.     {
104.         sb.append(String.format("%02X", b[i] & 0xFF));
105.     }
106.     return sb.toString();
107. }
108.
109. public static void main(String[] args)
110. {
111.     String[] testStrings = { "", "Sanfoundry", "Message Digest",
112.         "abcdefghijklmnopqrstuvwxyz" };
113.     for (String s : testStrings)
114.         System.out.println("0x" + toHexString(computeMD5(s.getBytes())))
115.             + " <== \" " + s + " \"");
116.     return;
117. }

```

118. }

Output:

advertisement

```
$ javac MD5.java  
$ java MD5
```

```
0xD41D8CD98F00B204E9800998ECF8427E <== ""  
0x123EC1617559F98A4C86AF629FEF21E6 <== "Sanfoundry"  
0xBBD9D8CC4AD8AD2599DBF623E7E5282E <== "Message Digest"  
0xC3FCD3D76192E4007DFB496CCA67E13B <== "abcdefghijklmnopqrstuvwxyz"
```

353.Java Program to Implement the One Time Pad Algorithm

[« Prev](#)

[Next »](#)

This is a java program to implement one time pad algorithm. In cryptography, a one-time pad (OTP) is an encryption technique that cannot be cracked if used correctly. In this technique, a plaintext is paired with random, secret key (or pad). Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition. If the key is truly random, and at least as long as the plaintext, and never reused in whole or in part, and kept completely secret, then the resulting ciphertext will be impossible to decrypt or break. It has also been proven that any cipher with the perfect secrecy property must use keys with effectively the same requirements as OTP keys. However, practical problems have prevented one-time pads from being widely used.

Here is the source code of the Java Program to Implement the One Time Pad Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class OneTimePadCipher
7. {
8.     public static String encryptionMessage(String s)
9.     {
10.         int i, j;
11.         int randomBitPattern[] = new int[8];
12.         for (i = 0; i < 7; i++)
13.         {
14.             randomBitPattern[i] = (i % 2 == 0) ? 1 : 0;
15.         }
16.         char asc[] = new char[s.length()];
17.         for (i = 0; i < s.length(); i++)
18.         {
19.             asc[i] = (char) ((int) s.charAt(i));
20.         }
21.         BasicOperation b1 = new BasicOperation();
22.         String cipherText = new String("");
23.         for (i = 0; i < asc.length; i++)
24.         {
25.             int temp = (int) (asc[i]);
26.             int len = b1.decimalToBinary(temp);
27.             int bintemp[] = new int[7];
28.             int xorlen;
29.             if (len == 7)
30.             {
31.                 for (j = 1; j <= len; j++)
32.                 {
33.                     bintemp[j - 1] = b1.binaryArrayAtPosition(j);
34.                 }
35.                 // XOR Operation
36.                 xorlen = b1.xorop(bintemp, randomBitPattern, len);
37.             }
38.             else
39.             {
40.                 // System.out.println("\n less than 7 :" + len);
41.                 bintemp[0] = 0;
42.                 for (j = 1; j <= len; j++)
43.                 {
44.                     bintemp[j] = b1.binaryArrayAtPosition(j);
```

```

45.        }
46.        // XOR Operation
47.        xorlen = b1.xorop(bintemp, randomBitPattern, len + 1);
48.    }
49.    int xor[] = new int[xorlen];
50.    for (j = 0; j < xorlen; j++)
51.    {
52.        xor[j] = b1.xorinArrayAt(j);
53.        cipherText = cipherText + xor[j];
54.    }
55.    cipherText += " ";
56. }
57. return (cipherText);
58. }
59.
60. public static String decryptionMessage(String s)
61. {
62.     int i, j;
63.     // char cipherChar[] = new char[s.length() / 2];
64.     char cipherChar[] = new char[(s.length())];
65.     int cnt = -1;
66.     for (i = 0; i < s.length(); i++)
67.     {
68.         // we receive only Ascii of it is allow 0 and 1, do not accept white
69.         // space
70.         // int ascii = (int)s.charAt(i);
71.         if ((int) s.charAt(i) == 48 || (int) s.charAt(i) == 49
72.             || (int) s.charAt(i) == 32)
73.         {
74.             cnt++;
75.             cipherChar[cnt] = s.charAt(i);
76.         }
77.     }
78.     String s1 = new String(cipherChar);
79.     String s2[] = s1.split(" ");
80.     int data[] = new int[s2.length];
81.     for (i = 0; i < s2.length; i++)
82.     {
83.         data[i] = Integer.parseInt(s2[i]);
84.     }
85.     char randomBitPattern[] = new char[7];
86.     for (i = 0; i < 7; i++)
87.     {
88.         randomBitPattern[i] = (i % 2 == 0) ? '1' : '0';
89.     }
90.     BasicOperation b1 = new BasicOperation();
91.     String plain = new String("");
92.     // do the XOR Operation
93.     for (i = 0; i < s2.length; i++)
94.     {
95.         int xorlen = b1.xorop(s2[i], randomBitPattern);
96.         int xor[] = new int[xorlen];
97.         for (j = 0; j < xorlen; j++)
98.         {
99.             xor[j] = b1.xorinArrayAt(j);
100.            plain += xor[j];
101.        }
102.        plain += " ";
103.    }
104.    String p[] = plain.split(" ");
105.    BasicOperation ob = new BasicOperation();
106.    int decryptedChar[] = new int[p.length];
107.    char plainTextChar[] = new char[p.length];
108.    for (i = 0; i < p.length; i++)
109.    {
110.        decryptedChar[i] = ob.binaryToDecimal(Integer.parseInt(p[i]));

```

```

111.     plainTextChar[i] = (char) decryptedChar[i];
112. }
113. return (new String(plainTextChar));
114. }
115.
116. public static void main(String[] args)
117. {
118.     Scanner sc = new Scanner(System.in);
119.     System.out.println("Enter the message: ");
120.     String message = sc.nextLine();
121.     System.out.println("'" + message + "' in encrypted message : "
122.                         + encryptionMessage(message));
123.     System.out.println("'" + encryptionMessage(message)
124.                         + "' in decrypted message : "
125.                         + decryptionMessage(encryptionMessage(message)));
126.     sc.close();
127. }
128. }
129.
130. class BasicOperation
131. {
132.     int bin[] = new int[100];
133.     int xor[] = new int[100];
134.     int temp1[] = new int[100];
135.     int temp2[] = new int[100];
136.     int len;
137.     int xorlen;
138.
139. // convert binary number to decimal number
140. public int binaryToDecimal(int myNum)
141. {
142.     int dec = 0, no, i, n = 0;
143.     no = myNum;
144.     // Find total digit of no of inupted number
145.     while (no > 0)
146.     {
147.         n++;
148.         no = no / 10;
149.     }
150.     // Convert inupted number into decimal
151.     no = myNum;
152.     for (i = 0; i < n; i++)
153.     {
154.         int temp = no % 10;
155.         dec = dec + temp * ((int) Math.pow(2, i));
156.         no = no / 10;
157.     }
158.     return dec;
159. }
160.
161. public int decimalToBinary(int myNum)
162. {
163.     int j, i = -1, no, temp = 0;
164.     no = myNum;
165.     int t[] = new int[100];
166.     while (no > 0)
167.     {
168.         i++;
169.         temp = no % 2;
170.         t[i] = temp;
171.         no = no / 2;
172.     }
173.     len = (i + 1);
174.     j = -1;
175.     for (i = len; i >= 0; i--)
176.     {

```

```

177.     j++;
178.     bin[j] = t[i];
179. }
180. return len;
181. }
182.
183. //find the specific bit value of binary number at given position
184. public int binaryArrayAtPosition(int pos)
185. {
186.     return bin[pos];
187. }
188.
189. public int xorinArrayAt(int pos)
190. {
191.     return xor[pos];
192. }
193.
194. //perform the binary X-OR operation
195. public int xorop(int a[], int b[], int arrlen)
196. {
197.     int i;
198.     for (i = 0; i < arrlen; i++)
199.     {
200.         xor[i] = (a[i] == b[i]) ? 0 : 1;
201.     }
202.     xorlen = i;
203.     return xorlen;
204. }
205.
206. //perform the binary X-OR operation
207. public int xorop(String s, char c[])
208. {
209.     int i = -1;
210.     for (i = 0; i < s.length(); i++)
211.     {
212.         xor[i] = (s.charAt(i) == c[i]) ? 0 : 1;
213.     }
214.     xorlen = i;
215.     return xorlen;
216. }
217.
218. public int getLen()
219. {
220.     return len + 1;
221. }
222.
223. //display binary bit pattern or the array
224. public void displayBinaryArray()
225. {
226.     for (int i = 0; i <= len; i++)
227.     {
228.         System.out.println("\n Binary Array :" + bin[i]);
229.     }
230. }
231. }

```

Output:

advertisement

```
$ javac OneTimePadCipher.java
$ java OneTimePadCipher
```

Enter the message:

Sanfoundry

'Sanfoundry' in encrypted message : 0000110 0110100 0111011 0110011 0111010 0100000 0111011 0110001 0100111 0101100
 '0000110 0110100 0111011 0110011 0111010 0100000 0111011 0110001 0100111 0101100' in decrypted message : Sa

354. Java Program to Decode a Message Encoded Using Playfair Cipher

[« Prev](#)

[Next »](#)

This is a java program to implement playfair cipher algorithm. The Playfair cipher or Playfair square is a manual symmetric encryption technique and was the first literal digraph substitution cipher.

Here is the source code of the Java Program to Decode a Message Encoded Using Playfair Cipher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class PlayfairCipherDecryption
7. {
8.     private String KeyWord      = new String();
9.     private String Key         = new String();
10.    private char  matrix_arr[][] = new char[5][5];
11.
12.   public void setKey(String k)
13.   {
14.       String K_adjust = new String();
15.       boolean flag = false;
16.       K_adjust = K_adjust + k.charAt(0);
17.       for (int i = 1; i < k.length(); i++)
18.       {
19.           for (int j = 0; j < K_adjust.length(); j++)
20.           {
21.               if (k.charAt(i) == K_adjust.charAt(j))
22.               {
23.                   flag = true;
24.               }
25.           }
26.           if (flag == false)
27.               K_adjust = K_adjust + k.charAt(i);
28.           flag = false;
29.       }
30.       KeyWord = K_adjust;
31.   }
32.
33.   public void KeyGen()
34.   {
35.       boolean flag = true;
36.       char current;
37.       Key = KeyWord;
38.       for (int i = 0; i < 26; i++)
39.       {
40.           current = (char) (i + 97);
41.           if (current == 'j')
42.               continue;
43.           for (int j = 0; j < KeyWord.length(); j++)
44.           {
45.               if (current == KeyWord.charAt(j))
46.               {
47.                   flag = false;
48.                   break;
49.               }
50.           }
51.           if (flag)
```

```

52.         Key = Key + current;
53.         flag = true;
54.     }
55.     System.out.println(Key);
56.     matrix();
57. }
58.
59. private void matrix()
60. {
61.     int counter = 0;
62.     for (int i = 0; i < 5; i++)
63.     {
64.         for (int j = 0; j < 5; j++)
65.         {
66.             matrix_arr[i][j] = Key.charAt(counter);
67.             System.out.print(matrix_arr[i][j] + " ");
68.             counter++;
69.         }
70.         System.out.println();
71.     }
72. }
73.
74. private String format(String old_text)
75. {
76.     int i = 0;
77.     int len = 0;
78.     String text = new String();
79.     len = old_text.length();
80.     for (int tmp = 0; tmp < len; tmp++)
81.     {
82.         if (old_text.charAt(tmp) == 'j')
83.         {
84.             text = text + 'i';
85.         }
86.         else
87.             text = text + old_text.charAt(tmp);
88.     }
89.     len = text.length();
90.     for (i = 0; i < len; i = i + 2)
91.     {
92.         if (text.charAt(i + 1) == text.charAt(i))
93.         {
94.             text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
95.         }
96.     }
97.     return text;
98. }
99.
100. private String[] Divid2Pairs(String new_string)
101. {
102.     String Original = format(new_string);
103.     int size = Original.length();
104.     if (size % 2 != 0)
105.     {
106.         size++;
107.         Original = Original + 'x';
108.     }
109.     String x[] = new String[size / 2];
110.     int counter = 0;
111.     for (int i = 0; i < size / 2; i++)
112.     {
113.         x[i] = Original.substring(counter, counter + 2);
114.         counter = counter + 2;
115.     }
116.     return x;
117. }

```

```

118.
119. public int[] GetDiminsions(char letter)
120. {
121.     int[] key = new int[2];
122.     if (letter == 'j')
123.         letter = 'i';
124.     for (int i = 0; i < 5; i++)
125.     {
126.         for (int j = 0; j < 5; j++)
127.         {
128.             if (matrix_arr[i][j] == letter)
129.             {
130.                 key[0] = i;
131.                 key[1] = j;
132.                 break;
133.             }
134.         }
135.     }
136.     return key;
137. }
138.
139. public String encryptMessage(String Source)
140. {
141.     String src_arr[] = Divid2Pairs(Source);
142.     String Code = new String();
143.     char one;
144.     char two;
145.     int part1[] = new int[2];
146.     int part2[] = new int[2];
147.     for (int i = 0; i < src_arr.length; i++)
148.     {
149.         one = src_arr[i].charAt(0);
150.         two = src_arr[i].charAt(1);
151.         part1 = GetDiminsions(one);
152.         part2 = GetDiminsions(two);
153.         if (part1[0] == part2[0])
154.         {
155.             if (part1[1] < 4)
156.                 part1[1]++;
157.             else
158.                 part1[1] = 0;
159.             if (part2[1] < 4)
160.                 part2[1]++;
161.             else
162.                 part2[1] = 0;
163.         }
164.         else if (part1[1] == part2[1])
165.         {
166.             if (part1[0] < 4)
167.                 part1[0]++;
168.             else
169.                 part1[0] = 0;
170.             if (part2[0] < 4)
171.                 part2[0]++;
172.             else
173.                 part2[0] = 0;
174.         }
175.         else
176.         {
177.             int temp = part1[1];
178.             part1[1] = part2[1];
179.             part2[1] = temp;
180.         }
181.         Code = Code + matrix_arr[part1[0]][part1[1]]
182.             + matrix_arr[part2[0]][part2[1]];
183.     }

```

```

184.     return Code;
185. }
186.
187. public String decryptMessage(String Code)
188. {
189.     String Original = new String();
190.     String src_arr[] = Divid2Pairs(Code);
191.     char one;
192.     char two;
193.     int part1[] = new int[2];
194.     int part2[] = new int[2];
195.     for (int i = 0; i < src_arr.length; i++)
196.     {
197.         one = src_arr[i].charAt(0);
198.         two = src_arr[i].charAt(1);
199.         part1 = GetDimensions(one);
200.         part2 = GetDimensions(two);
201.         if (part1[0] == part2[0])
202.         {
203.             if (part1[1] > 0)
204.                 part1[1]--;
205.             else
206.                 part1[1] = 4;
207.             if (part2[1] > 0)
208.                 part2[1]--;
209.             else
210.                 part2[1] = 4;
211.         }
212.         else if (part1[1] == part2[1])
213.         {
214.             if (part1[0] > 0)
215.                 part1[0]--;
216.             else
217.                 part1[0] = 4;
218.             if (part2[0] > 0)
219.                 part2[0]--;
220.             else
221.                 part2[0] = 4;
222.         }
223.         else
224.         {
225.             int temp = part1[1];
226.             part1[1] = part2[1];
227.             part2[1] = temp;
228.         }
229.         Original = Original + matrix_arr[part1[0]][part1[1]]
230.             + matrix_arr[part2[0]][part2[1]];
231.     }
232.     return Original;
233. }
234.
235. public static void main(String[] args)
236. {
237.     PlayfairCipherDecryption x = new PlayfairCipherDecryption();
238.     Scanner sc = new Scanner(System.in);
239.     System.out.println("Enter a keyword:");
240.     String keyword = sc.next();
241.     x.setKey(keyword);
242.     x.KeyGen();
243.     System.out
244.         .println("Enter word to encrypt: (Make sure length of message is even)");
245.     String key_input = sc.next();
246.     if (key_input.length() % 2 == 0)
247.     {
248.         System.out.println("Encryption: " + x.encryptMessage(key_input));
249.         System.out.println("Decryption: "

```

```
250.          + x.decryptMessage(x.encryptMessage(key_input)));
251.      }
252.  else
253.  {
254.      System.out.println("Message length should be even");
255.  }
256. sc.close();
257. }
258.}
```

Output:

advertisement

```
$ javac PlayfairCipherDecryption.java
$ java PlayfairCipherDecryption
```

Enter a keyword:

```
sanfoundry
sanfoudrybceghiklmqpqtvwxz
s a n f o
u d r y b
c e g h i
k l m p q
t v w x z
```

Enter word to encrypt: (Make sure length of message is even)

```
learning
```

```
Encryption: vlndogrm
```

```
Decryption: learning
```

355.Java Program to Encode a Message Using Playfair Cipher

[« Prev](#)

[Next »](#)

This is a java program to implement playfair cipher algorithm. The Playfair cipher or Playfair square is a manual symmetric encryption technique and was the first literal digraph substitution cipher.

Here is the source code of the Java Program to Enode a Message Using Playfair Cipher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class PlayfairCipherEncryption
7. {
8.     private String KeyWord      = new String();
9.     private String Key         = new String();
10.    private char  matrix_arr[][] = new char[5][5];
11.
12.   public void setKey(String k)
13.   {
14.       String K_adjust = new String();
15.       boolean flag = false;
16.       K_adjust = K_adjust + k.charAt(0);
17.       for (int i = 1; i < k.length(); i++)
18.       {
19.           for (int j = 0; j < K_adjust.length(); j++)
20.           {
21.               if (k.charAt(i) == K_adjust.charAt(j))
22.               {
23.                   flag = true;
24.               }
25.           }
26.           if (flag == false)
27.               K_adjust = K_adjust + k.charAt(i);
28.           flag = false;
29.       }
30.       KeyWord = K_adjust;
31.   }
32.
33.   public void KeyGen()
34.   {
35.       boolean flag = true;
36.       char current;
37.       Key = KeyWord;
38.       for (int i = 0; i < 26; i++)
39.       {
40.           current = (char) (i + 97);
41.           if (current == 'j')
42.               continue;
43.           for (int j = 0; j < KeyWord.length(); j++)
44.           {
45.               if (current == KeyWord.charAt(j))
46.               {
47.                   flag = false;
48.                   break;
49.               }
50.           }
51.           if (flag)
```

```

52.         Key = Key + current;
53.         flag = true;
54.     }
55.     System.out.println(Key);
56.     matrix();
57. }
58.
59. private void matrix()
60. {
61.     int counter = 0;
62.     for (int i = 0; i < 5; i++)
63.     {
64.         for (int j = 0; j < 5; j++)
65.         {
66.             matrix_arr[i][j] = Key.charAt(counter);
67.             System.out.print(matrix_arr[i][j] + " ");
68.             counter++;
69.         }
70.         System.out.println();
71.     }
72. }
73.
74. private String format(String old_text)
75. {
76.     int i = 0;
77.     int len = 0;
78.     String text = new String();
79.     len = old_text.length();
80.     for (int tmp = 0; tmp < len; tmp++)
81.     {
82.         if (old_text.charAt(tmp) == 'j')
83.         {
84.             text = text + 'i';
85.         }
86.         else
87.             text = text + old_text.charAt(tmp);
88.     }
89.     len = text.length();
90.     for (i = 0; i < len; i = i + 2)
91.     {
92.         if (text.charAt(i + 1) == text.charAt(i))
93.         {
94.             text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
95.         }
96.     }
97.     return text;
98. }
99.
100. private String[] Divid2Pairs(String new_string)
101. {
102.     String Original = format(new_string);
103.     int size = Original.length();
104.     if (size % 2 != 0)
105.     {
106.         size++;
107.         Original = Original + 'x';
108.     }
109.     String x[] = new String[size / 2];
110.     int counter = 0;
111.     for (int i = 0; i < size / 2; i++)
112.     {
113.         x[i] = Original.substring(counter, counter + 2);
114.         counter = counter + 2;
115.     }
116.     return x;
117. }

```

```

118.
119. public int[] GetDiminsions(char letter)
120. {
121.     int[] key = new int[2];
122.     if (letter == 'j')
123.         letter = 'i';
124.     for (int i = 0; i < 5; i++)
125.     {
126.         for (int j = 0; j < 5; j++)
127.         {
128.             if (matrix_arr[i][j] == letter)
129.             {
130.                 key[0] = i;
131.                 key[1] = j;
132.                 break;
133.             }
134.         }
135.     }
136.     return key;
137. }
138.
139. public String encryptMessage(String Source)
140. {
141.     String src_arr[] = Divid2Pairs(Source);
142.     String Code = new String();
143.     char one;
144.     char two;
145.     int part1[] = new int[2];
146.     int part2[] = new int[2];
147.     for (int i = 0; i < src_arr.length; i++)
148.     {
149.         one = src_arr[i].charAt(0);
150.         two = src_arr[i].charAt(1);
151.         part1 = GetDiminsions(one);
152.         part2 = GetDiminsions(two);
153.         if (part1[0] == part2[0])
154.         {
155.             if (part1[1] < 4)
156.                 part1[1]++;
157.             else
158.                 part1[1] = 0;
159.             if (part2[1] < 4)
160.                 part2[1]++;
161.             else
162.                 part2[1] = 0;
163.         }
164.         else if (part1[1] == part2[1])
165.         {
166.             if (part1[0] < 4)
167.                 part1[0]++;
168.             else
169.                 part1[0] = 0;
170.             if (part2[0] < 4)
171.                 part2[0]++;
172.             else
173.                 part2[0] = 0;
174.         }
175.         else
176.         {
177.             int temp = part1[1];
178.             part1[1] = part2[1];
179.             part2[1] = temp;
180.         }
181.         Code = Code + matrix_arr[part1[0]][part1[1]]
182.             + matrix_arr[part2[0]][part2[1]];
183.     }

```

```
184.     return Code;
185. }
186.
187. public static void main(String[] args)
188. {
189.     PlayfairCipherEncryption x = new PlayfairCipherEncryption();
190.     Scanner sc = new Scanner(System.in);
191.     System.out.println("Enter a keyword:");
192.     String keyword = sc.next();
193.     x.setKey(keyword);
194.     x.KeyGen();
195.     System.out
196.         .println("Enter word to encrypt: (Make sure length of message is even)");
197.     String key_input = sc.next();
198.     if (key_input.length() % 2 == 0)
199.     {
200.         System.out.println("Encryption: " + x.encryptMessage(key_input));
201.     }
202.     else
203.     {
204.         System.out.println("Message length should be even");
205.     }
206.     sc.close();
207. }
208. }
```

Output:

advertisement

```
$ javac PlayfairCipherEncryption.java
$ java PlayfairCipherEncryption
```

```
Enter a keyword:
Sanfoundry
Sanfoudrybceghiklmqpstvwxz
S a n f o
u d r y b
c e g h i
k l m p q
s t v w x
Enter word to encrypt: (Make sure length of message is even)
Learningcenter
Encryption: acndogrmegavgd
```

Java Program to Implement Caesar Cypher

[« Prev](#)

[Next »](#)

This is a java program to implement Caesar Cipher Encryption algorithm. This is the simplest of all, where every character of the message is replaced by its next 3rd character.

Here is the source code of the Java Program to Implement Caesar Cypher. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class CaesarCipher
7. {
8.     public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
9.
10.    public static String encrypt(String plainText, int shiftKey)
11.    {
12.        plainText = plainText.toLowerCase();
13.        String cipherText = "";
14.        for (int i = 0; i < plainText.length(); i++)
15.        {
16.            int charPosition = ALPHABET.indexOf(plainText.charAt(i));
17.            int keyVal = (shiftKey + charPosition) % 26;
18.            char replaceVal = ALPHABET.charAt(keyVal);
19.            cipherText += replaceVal;
20.        }
21.        return cipherText;
22.    }
23.
24.    public static String decrypt(String cipherText, int shiftKey)
25.    {
26.        cipherText = cipherText.toLowerCase();
27.        String plainText = "";
28.        for (int i = 0; i < cipherText.length(); i++)
29.        {
30.            int charPosition = ALPHABET.indexOf(cipherText.charAt(i));
31.            int keyVal = (charPosition - shiftKey) % 26;
32.            if (keyVal < 0)
33.            {
34.                keyVal = ALPHABET.length() + keyVal;
35.            }
36.            char replaceVal = ALPHABET.charAt(keyVal);
37.            plainText += replaceVal;
38.        }
39.        return plainText;
40.    }
41.
42.    public static void main(String[] args)
43.    {
44.        Scanner sc = new Scanner(System.in);
45.        System.out.println("Enter the String for Encryption: ");
46.        String message = new String();
47.        message = sc.nextLine();
48.        System.out.println(encrypt(message, 3));
49.        System.out.println(decrypt(encrypt(message, 3), 3));
50.        sc.close();
51.    }
52.}
```

Output:

advertisement

```
$ javac CaesarCipher.java  
$ java CaesarCipher
```

Enter the String for Encryption:

```
Sanfoundry  
vdqirxqgub  
sanfoundry
```

356.Java Program to Implement Wagner and Fisher Algorithm for online String Matching

[« Prev](#)

[Next »](#)

This is a java program to implement Wagner and Fisher Algorithm. In computer science, the Wagner–Fischer algorithm is a dynamic programming algorithm that computes the edit distance between two strings of characters.

Here is the source code of the Java Program to Implement Wagner and Fisher Algorithm for online String Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. 
2. package com.sanfoundry.setandstring;
3. 
4. import java.io.BufferedReader;
5. import java.io.IOException;
6. import java.io.InputStreamReader;
7. 
8. public class WagnerandFischer
9. {
10.    public int getLevenshteinDistance(String str1, String str2)
11.    {
12.        int len1 = str1.length();
13.        int len2 = str2.length();
14.        int[][] arr = new int[len1 + 1][len2 + 1];
15.        for (int i = 0; i <= len1; i++)
16.            arr[i][0] = i;
17.        for (int i = 1; i <= len2; i++)
18.            arr[0][i] = i;
19.        for (int i = 1; i <= len1; i++)
20.        {
21.            for (int j = 1; j <= len2; j++)
22.            {
23.                int m = (str1.charAt(i - 1) == str2.charAt(j - 1)) ? 0 : 1;
24.                arr[i][j] = Math.min(
25.                    Math.min(arr[i - 1][j] + 1, arr[i][j - 1] + 1),
26.                    arr[i - 1][j - 1] + m);
27.            }
28.        }
29.        return arr[len1][len2];
30.    }
31. 
32. public static void main(String[] args) throws IOException
33. {
34.    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
35.    System.out.println("Enter string 1 :");
36.    String str1 = br.readLine();
37.    System.out.println("Enter string 2 :");
38.    String str2 = br.readLine();
39.    WagnerandFischer wf = new WagnerandFischer();
40.    int lDist = wf.getLevenshteinDistance(str1, str2);
41.    System.out.println("Edit (Levenshtein) Distance between two strings = "
42.        + lDist);
43.    br.close();
44. }
45. }
```

Output:

advertisement

\$ javac WagnerandFischer.java

```
$ java WagnerandFischer
```

Enter string 1 :

Wagner and Fisher Algorithm

Enter string 2 :

Sanfoundry is No. 1 choice for Deep Hands-ON Trainings in SAN, Linux & C, Kernel & Device Driver Programming. Our Founder has trained employees of almost all Top Companies in India. Here are few of them: VMware, Citrix, Oracle, Motorola, Ericsson, Aricent, HP, Intuit, Microsoft, Cisco, SAP Labs, Siemens, Symantec, Redhat, Chelsio, Cavium Networks, ST Microelectronics, Samsung, LG-Soft, Wipro, TCS, HCL, IBM, Accenture, HSBC, Northwest Bank, Mphasis, Tata Elxsi, Tata Communications, Mindtree, Cognizant, mid size IT companies and many Startups. Students from top Universities and colleges such as NIT Trichy, BITS Pilani, University of California, Irvine, University of Texas, Austin & PESIT Bangalore have benefited a lot from these courses as well. The assignments and real time projects for our courses are of extremely high quality with excellent learning curve.

Edit (Levenshtein) Distance between two strings = 844

357.Java Program to Implement Levenshtein Distance Computing Algorithm

[« Prev](#)

[Next »](#)

This is a java program to implement Levenshtein Distance Computing Algorithm. In computer science, edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of macromolecules such as DNA, which can be viewed as strings of the letters A, C, G and T. Several definitions of edit distance exist, using different sets of string operations. One of the most common variants is called Levenshtein distance.

Here is the source code of the Java Program to Implement Levenshtein Distance Computing Algorithm. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.ArrayList;
5. import java.util.Collections;
6. import java.util.List;
7. import java.util.Scanner;
8.
9. public class ApproxStringMatchingUsingLevenshteinDistance
10. {
11.     public static int distance(String a, String b)
12.     {
13.         a = a.toLowerCase();
14.         b = b.toLowerCase();
15.         int[] costs = new int[b.length() + 1];
16.         for (int j = 0; j < costs.length; j++)
17.             costs[j] = j;
18.         for (int i = 1; i <= a.length(); i++)
19.         {
20.             costs[0] = i;
21.             int nw = i - 1;
22.             for (int j = 1; j <= b.length(); j++)
23.             {
24.                 int cj = Math.min(1 + Math.min(costs[j], costs[j - 1]),
25.                     a.charAt(i - 1) == b.charAt(j - 1) ? nw : nw + 1);
26.                 nw = costs[j];
27.                 costs[j] = cj;
28.             }
29.         }
30.         return costs[b.length()];
31.     }
32.
33.     public static void main(String[] args)
34.     {
35.         Scanner sc = new Scanner(System.in);
36.         String text = "In computer science, approximate string matching, "
37.             + "often colloquially referred to as fuzzy string searching is the technique of finding strings that match a
38.             pattern approximately rather than exactly. "
39.             + "The problem of approximate string matching is typically divided into two sub-problems: finding
40.             approximate substring matches inside a given string and finding "
41.             + "dictionary strings that match the pattern approximately. ";
42.         System.out.println("System generated string is: \n" + text + "\n");
43.         System.out.println("Enter the keyword to search: ");
44.         String keyword = sc.nextLine();
```

```
43. String[] data = text.split(" ");
44. List<Integer> dist = new ArrayList<Integer>();
45. for (int i = 0; i < data.length; i++)
46. {
47.     dist.add(distance(data[i], keyword));
48. }
49. Collections.sort(dist);
50. System.out.print("Did you mean: ");
51. for (int i = 0; i < data.length; i++)
52. {
53.     if (distance(data[i], keyword) == dist.get(0))
54.     {
55.         System.out.print(data[i] + " ");
56.     }
57. }
58. sc.close();
59. }
60. }
```

Output:

advertisement

```
$ javac ApproxStringMatchingUsingLevenshteinDistance.java
$ java ApproxStringMatchingUsingLevenshteinDistance
```

System generated string is:

'In computer science, approximate string matching, often colloquially referred to as fuzzy string searching is the technique of finding strings that match a pattern approximately rather than exactly. The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately.'

Enter the keyword to search:

cpcputer

Did you mean: computer

358.Java Program to Use Dynamic Programming to Solve Approximate String Matching

[« Prev](#)

[Next »](#)

This is a java program to solve approximate string matching using dynamic programming.

Here is the source code of the Java Program to Use Dynamic Programming to Solve Approximate String Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.ArrayList;
5. import java.util.Collections;
6. import java.util.HashMap;
7. import java.util.List;
8. import java.util.Map;
9. import java.util.Scanner;
10.
11. public class ApproxStringMatching
12. {
13.     private List<String> foods = new ArrayList<String>();
14.     private Map<String, Double> matchingScores = new HashMap<String, Double>();
15.     private Scanner scanner;
16.     private static double[][] mismatchScoreTable;
17.     private String in;
18.     private int inLength;
19.
20.     public ApproxStringMatching(String text)
21.     {
22.         /*
23.          * read the file, fill the food list
24.          */
25.         try
26.         {
27.             scanner = new Scanner(text);
28.             while (scanner.hasNext())
29.             {
30.                 this.foods.add(scanner.nextLine());
31.             }
32.         }
33.         catch (Exception e)
34.         {
35.             e.printStackTrace();
36.             System.exit(1);
37.         }
38.         if (mismatchScoreTable == null)
39.             initMismatchScoreTable();
40.     }
41.
42.     public List<String> getFoods()
43.     {
44.         return this.foods;
45.     }
46.
47.     private static void initMismatchScoreTable()
48.     {
49.         mismatchScoreTable = new double[256][256];
50.         /*
51.          * Score any combination of two characters as 1 by default.
52.          */
```

```

53.     for (int i = 0; i < 256; i++)
54.         for (int j = 0; j < 256; j++)
55.             mismatchScoreTable[i][j] = 1.0d;
56.     /*
57.      * If the input character and reference character are the same,
58.      * there is no typo. So the error score is 0.
59.      */
60.     for (int i = 0; i < 256; i++)
61.         mismatchScoreTable[i][i] = 0.0d;
62.     /*
63.      * For people who use both German keyboard and English keyboard,
64.      * this typo is highly frequent.
65.      */
66.     mismatchScoreTable['y']['z'] = 0.1d;
67.     mismatchScoreTable['z']['y'] = 0.1d;
68.     mismatchScoreTable['v']['b'] = 0.15d;
69.     mismatchScoreTable['b']['v'] = 0.15d;
70.     mismatchScoreTable['n']['m'] = 0.11d;
71.     mismatchScoreTable['m']['n'] = 0.11d;
72.     mismatchScoreTable['t']['r'] = 0.15d;
73.     mismatchScoreTable['r']['t'] = 0.15d;
74.     mismatchScoreTable['g']['h'] = 0.15d;
75.     mismatchScoreTable['h']['g'] = 0.15d;
76.     mismatchScoreTable['y']['u'] = 0.15d;
77.     mismatchScoreTable['u']['y'] = 0.15d;
78.     /*
79.      * more typo possibilities can be inserted here....
80.      */
81. }
82.
83. public Map<String, Double> getScores(String in)
84. {
85.     this.in = in;
86.     this.inLength = in.length();
87.     for (String food : this.foods)
88.     {
89.         int refLength = food.length();
90.         double[][] errScore = new double[inLength + 1][refLength + 1];
91.         errScore[0][0] = 0.0d;
92.         for (int inCharAt = 1; inCharAt <= this.inLength; inCharAt++)
93.             errScore[inCharAt][0] = inCharAt;
94.         for (int refCharAt = 1; refCharAt <= refLength; refCharAt++)
95.             errScore[0][refCharAt] = refCharAt;
96.         for (int inCharAt = 1; inCharAt <= inLength; inCharAt++)
97.             for (int refCharAt = 1; refCharAt <= refLength; refCharAt++)
98.             {
99.                 /*
100.                  * if a character is absent at the given position
101.                  * in the input string, we add score 1.
102.                  */
103.                 double charAbsence = errScore[inCharAt - 1][refCharAt] + 1;
104.                 /*
105.                  * if a character is redundant at the given position in the
106.                  * input string, we add score 1.
107.                  */
108.                 double charRedundance = errScore[inCharAt][refCharAt - 1] + 1;
109.                 /*
110.                  * if it is a matching error, we add the score specified in
111.                  * the score table for matching errors.
112.                  */
113.                 double mismatch = errScore[inCharAt - 1][refCharAt - 1]
114.                     + mismatchScoreTable[this.in.charAt(inCharAt - 1)][food
115.                         .charAt(refCharAt - 1)];
116.                 /*
117.                  * initialize the score for swap error to a very big value.
118.                  */

```

```

119.         double charPositionSwap = 999999d;
120.         /*
121.          * score for swap error
122.          */
123.         if (inCharAt > 1
124.             && refCharAt > 1
125.             && this.in.charAt(inCharAt - 1) == food
126.                 .charAt(refCharAt - 2)
127.             && this.in.charAt(inCharAt - 2) == food
128.                 .charAt(refCharAt - 1))
129.         {
130.             /*
131.              * the score for typing "ie" as "ei" and vice versa
132.              * is even lower
133.              */
134.             if (this.in.charAt(inCharAt - 2) == 'e'
135.                 && this.in.charAt(inCharAt - 1) == 'i')
136.             {
137.                 charPositionSwap = errScore[inCharAt - 2][refCharAt - 2] + 0.25;
138.             }
139.             /*
140.               * more cases can be inserted here.
141.               */
142.             else
143.                 charPositionSwap = errScore[inCharAt - 2][refCharAt - 2] + 0.5;
144.         }
145.         /*
146.           * more error cases can be inserted here.
147.           */
148.         double minScore = mismatch;
149.         if (charAbsence < minScore)
150.         {
151.             minScore = charAbsence;
152.         }
153.         if (charRedundance < minScore)
154.         {
155.             minScore = charRedundance;
156.         }
157.         if (charPositionSwap < minScore)
158.         {
159.             minScore = charPositionSwap;
160.         }
161.         errScore[inCharAt][refCharAt] = minScore;
162.     }
163.     this.matchingScores.put(food, errScore[this.inLength][refLength]);
164. }
165. return this.matchingScores;
166. }
167.
168. @SuppressWarnings({ "unchecked", "rawtypes" })
169. public static void main(String[] args)
170. {
171.     String text = "In computer science, approximate string matching "
172.                 + "(often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a
173.                 pattern approximately (rather than exactly). "
174.                 + "The problem of approximate string matching is typically divided into two sub-problems: finding
175.                 approximate substring matches inside a given string and finding
176.                 + \"dictionary strings that match the pattern approximately.\">";
177.     Scanner sc = new Scanner(System.in);
178.     ApproxStringMatching demo = new ApproxStringMatching(text);
179.     System.out.print("Please type a word. Type q for exit: ");
180.     sc.nextLine();
181.     while (sc.hasNext())
182.     {
183.         String in = sc.nextLine();
184.         if (in.equals("q"))

```

```

183.     {
184.         System.exit(0);
185.     }
186.     System.out.println("You typed " + in);
187.     System.out.println("-----");
188.     Map scoreMap = demo.getScores(in);
189.     for (String food : demo.getFoods())
190.     {
191.         System.out.println(food + "\t error score: "
192.                             + scoreMap.get(food));
193.     }
194.     System.out.println("-----");
195.     double minScore = (Double) Collections.min(scoreMap.values());
196.     if (minScore == 0.0d)
197.     {
198.         System.out.println(in + " is in the list.");
199.     }
200.     else
201.     {
202.         List<String> corrections = new ArrayList<String>();
203.         StringBuffer sb = new StringBuffer("Do you mean:- ");
204.         for (String food : demo.getFoods())
205.         {
206.             if (scoreMap.get(food).equals(minScore))
207.             {
208.                 corrections.add(food);
209.                 sb.append(food).append(" or ");
210.             }
211.         }
212.         sb.append("?");
213.         System.out.println(sb.toString());
214.     }
215.     System.out.println("Please type a word. Type q for exit: ");
216. }
217. sc.close();
218. }
219. }
```

Output:

advertisement

```
$ javac ApproxStringMatching.java
$ java ApproxStringMatching
```

Please type a word. Type q for exit: String
You typed String

In computer science, approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately. error score: 417.0

Do you mean:- In computer science, approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately. or ?

Please type a word. Type q for exit: Matching
You typed Matching

In computer science, approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately. error score: 410.0

Do you mean:- In computer science, approximate string matching (often colloquially referred to as fuzzy string searching) is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching

is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately. or ?

Please type a word. Type q for exit:

359.Java Program to Implement String Matching Using Vectors

[« Prev](#)

[Next »](#)

This is a java program to perform search using Vectors.

Here is the source code of the Java Program to Implement String Matching Using Vectors. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5. import java.util.Vector;
6.
7. public class StringSearchUsingVectors
8. {
9.     public static void main(String[] args)
10.    {
11.        Scanner sc = new Scanner(System.in);
12.        System.out.println("Enter the main string: ");
13.        Vector<String> text = new Vector<String>();
14.        text.add(sc.nextLine());
15.        System.out.println("Enter the pattern string: ");
16.        Vector<String> pattern = new Vector<String>();
17.        pattern.add(sc.nextLine());
18.        for (int i = 0; i <= text.elementAt(0).length()
19.             - pattern.elementAt(0).length(); i++)
20.        {
21.            if (text.elementAt(0)
22.                .substring(i, i + pattern.elementAt(0).length())
23.                .equalsIgnoreCase(pattern.elementAt(0)))
24.            {
25.                System.out.println(pattern.elementAt(0)
26.                    + " is substring of main string, match found at: "
27.                    + ++i);
28.            }
29.        }
30.        sc.close();
31.    }
32.}
```

Output:

advertisement

```
$ javac StringSearchUsingVectors.java
$ java StringSearchUsingVectors
```

```
Enter the main string:
Java Program to Implement String Matching Using Vectors
Enter the pattern string:
Vectors
Vectors is substring of main string, match found at: 49
```

360.Java Program to Perform String Matching Using String Library

[« Prev](#)

[Next »](#)

This is a java program to perform search using string library.

Here is the source code of the Java Program to Perform String Matching Using String Library. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class StringSearchUsingStrLib
7. {
8.     public static void main(String[] args)
9.     {
10.         Scanner sc = new Scanner(System.in);
11.         System.out.println("Enter the main string: ");
12.         String text = sc.nextLine();
13.         System.out.println("Enter the pattern string: ");
14.         String pattern = sc.nextLine();
15.         for (int i = 0; i <= (text.length() - pattern.length()); i++)
16.         {
17.             if (text.substring(i, (i + pattern.length())).equalsIgnoreCase(
18.                 pattern))
19.             {
20.                 System.out.println(pattern
21.                     + " is substring of main string, match found at: "
22.                     + ++i);
23.             }
24.         }
25.         sc.close();
26.     }
27. }
```

Output:

advertisement

```
$ javac StringSearchUsingStrLib.java
$ java StringSearchUsingStrLib
```

```
Enter the main string:
Java Program to Perform String Matching Using String Library
Enter the pattern string:
String
String is substring of main string, match found at: 25
String is substring of main string, match found at: 47
```

361.Java Program to Perform String Matching Using String Library

[« Prev](#)

[Next »](#)

This is a java program to perform search using string library.

Here is the source code of the Java Program to Perform String Matching Using String Library. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class StringSearchUsingStrLib
7. {
8.     public static void main(String[] args)
9.     {
10.         Scanner sc = new Scanner(System.in);
11.         System.out.println("Enter the main string: ");
12.         String text = sc.nextLine();
13.         System.out.println("Enter the pattern string: ");
14.         String pattern = sc.nextLine();
15.         for (int i = 0; i <= (text.length() - pattern.length()); i++)
16.         {
17.             if (text.substring(i, (i + pattern.length())).equalsIgnoreCase(
18.                 pattern))
19.             {
20.                 System.out.println(pattern
21.                     + " is substring of main string, match found at: "
22.                     + ++i);
23.             }
24.         }
25.         sc.close();
26.     }
27. }
```

Output:

advertisement

```
$ javac StringSearchUsingStrLib.java
$ java StringSearchUsingStrLib
```

```
Enter the main string:
Java Program to Perform String Matching Using String Library
Enter the pattern string:
String
String is substring of main string, match found at: 25
String is substring of main string, match found at: 47
```

362.Java Program to Perform Finite State Automaton based Search

[« Prev](#)

[Next »](#)

This is a java program to perform search using DFA.

Here is the source code of the Java Program to Perform Finite State Automaton based Search. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. public class searchStringUsingDFA
7. {
8.     public static final int NO_OF_CHARS = 256;
9.
10.    public static int getNextState(char[] pat, int M, int state, int x)
11.    {
12.        /*
13.         * If the character c is same as next character in pattern,
14.         * then simply increment state
15.         */
16.        if (state < M && x == pat[state])
17.            return state + 1;
18.        int ns, i;
19.        /*
20.         * ns stores the result which is next state
21.         * ns finally contains the longest prefix which is also suffix
22.         * in "pat[0..state-1]c"
23.         * Start from the largest possible value and stop when you find
24.         * a prefix which is also suffix
25.         */
26.        for (ns = state; ns > 0; ns--)
27.        {
28.            if (pat[ns - 1] == x)
29.            {
30.                for (i = 0; i < ns - 1; i++)
31.                {
32.                    if (pat[i] != pat[state - ns + 1 + i])
33.                        break;
34.                }
35.                if (i == ns - 1)
36.                    return ns;
37.            }
38.        }
39.        return 0;
40.    }
41.
42. /*
43.  * This function builds the TF table which represents Finite Automata for a
44.  * given pattern
45. */
46. public static void computeTF(char[] pat, int M, int[][] TF)
47. {
48.     int state, x;
49.     for (state = 0; state <= M; ++state)
50.         for (x = 0; x < NO_OF_CHARS; ++x)
51.             TF[state][x] = getNextState(pat, M, state, x);
52. }
```

```

54. /*
55. * Prints all occurrences of pat in txt
56. */
57. public static void search(char[] pat, char[] txt)
58. {
59.     int M = pat.length;
60.     int N = txt.length;
61.     int[][] TF = new int[M + 1][NO_OF_CHARS];
62.     computeTF(pat, M, TF);
63.     //Process txt over FA.
64.     int i, state = 0;
65.     for (i = 0; i < N; i++)
66.     {
67.         state = TF[state][txt[i]];
68.         if (state == M)
69.         {
70.             System.out.print(pat);
71.             System.out.print(" found at " + (i - M + 1));
72.         }
73.     }
74. }
75.
76. public static void main(String[] args)
77. {
78.     Scanner sc = new Scanner(System.in);
79.     System.out.println("Enter the main string: ");
80.     String main = sc.nextLine();
81.     System.out.println("Enter the pattern string: ");
82.     String pattern = sc.nextLine();
83.     search(pattern.toCharArray(), main.toCharArray());
84.     sc.close();
85. }
86.

```

Output:

advertisement

```
$ javac searchStringUsingDFA.java
$ java searchStringUsingDFA
```

Enter the main string:

Sanfoundry is No. 1 choice for Deep Hands-ON Trainings in SAN, Linux & C, Kernel & Device Driver Programming. Our Founder has trained employees of almost all Top Companies in India. Here are few of them: VMware, Citrix, Oracle, Motorola, Ericsson, Aricent, HP, Intuit, Microsoft, Cisco, SAP Labs, Siemens, Symantec, Redhat, Chelsio, Cavium Networks, ST Microelectronics, Samsung, LG-Soft, Wipro, TCS, HCL, IBM, Accenture, HSBC, Northwest Bank, Mphasis, Tata Elxsi, Tata Communications, Mindtree, Cognizant, mid size IT companies and many Startups. Students from top Universities and colleges such as NIT Trichy, BITS Pilani, University of California, Irvine, University of Texas, Austin & PESIT Bangalore have benefited a lot from these courses as well. The assignments and real time projects for our courses are of extremely high quality with excellent learning curve.

Enter the pattern string:

Trainings

Trainings found at 45

363.Java Program to Perform Naive String Matching

« [Prev](#)

[Next](#) »

This is a java program to perform Naive String matching algorithm.

Here is the source code of the Java Program to Perform Naive String Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1. package com.sanfoundry.setandstring;
2.
3.
4. import java.util.Scanner;
5.
6. public class StringSearchUsingNaiveMethod
7. {
8.     private final int BASE;
9.     private int[] occurrence;
10.    private String pattern;
11.
12.    public StringSearchUsingNaiveMethod(String pattern)
13.    {
14.        this.BASE = 256;
15.        this.pattern = pattern;
16.        occurrence = new int[BASE];
17.        for (int c = 0; c < BASE; c++)
18.            occurrence[c] = -1;
19.        for (int j = 0; j < pattern.length(); j++)
20.            occurrence[pattern.charAt(j)] = j;
21.    }
22.
23.    public int search(String text)
24.    {
25.        int n = text.length();
26.        int m = pattern.length();
27.        int skip;
28.        for (int i = 0; i <= n - m; i += skip)
29.        {
30.            skip = 0;
31.            for (int j = m - 1; j >= 0; j--)
32.            {
33.                if (pattern.charAt(j) != text.charAt(i + j))
34.                {
35.                    skip = Math.max(1, j - occurrence[text.charAt(i + j)]);
36.                    break;
37.                }
38.            }
39.            if (skip == 0)
40.                return i;
41.        }
42.        return n;
43.    }
44.
45.    public static void main(String[] args)
46.    {
47.        Scanner sc = new Scanner(System.in);
48.        System.out.print("Enter the main string: ");
49.        String text = sc.nextLine();
50.        System.out.print("Enter the string to be searched: ");
51.        String pattern = sc.nextLine();
52.        StringSearchUsingNaiveMethod nsm = new StringSearchUsingNaiveMethod(
53.            pattern);
54.        int first_occur_position = nsm.search(text);
55.        System.out.println("The text " + pattern
```

```
56.         + "" is first found after the " + first_occur_position
57.         + " position.");
58.     sc.close();
59. }
60.}
```

Output:

advertisement

```
$ javac StringSearchUsingNaiveMethod.java
$ java StringSearchUsingNaiveMethod
```

Enter the main string: Sanfoundry is No. 1 choice for Deep Hands-ON Trainings in SAN, Linux & C, Kernel & Device Driver Programming. Our Founder has trained employees of almost all Top Companies in India. Here are few of them: VMware, Citrix, Oracle, Motorola, Ericsson, Aricent, HP, Intuit, Microsoft, Cisco, SAP Labs, Siemens, Symantec, Redhat, Chelsio, Cavium Networks, ST Microelectronics, Samsung, LG-Soft, Wipro, TCS, HCL, IBM, Accenture, HSBC, Northwest Bank, Mphasis, Tata Elxsi, Tata Communications, Mindtree, Cognizant, mid size IT companies and many Startups. Students from top Universities and colleges such as NIT Trichy, BITS Pilani, University of California, Irvine, University of Texas, Austin & PESIT Bangalore have benefited a lot from these courses as well. The assignments and real time projects for our courses are of extremely high quality with excellent learning curve.

Enter the string to be searched: No. 1

The text 'No. 1 ' is first found after the 14 position.

Java Program to Implement the Program Used in grep/egrep/fgrep

[« Prev](#)

[Next »](#)

This is a java program to implement grep linux command.

Here is the source code of the Java Program to Implement the Program Used in grep/egrep/fgrep. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.io.BufferedReader;
5. import java.io.FileReader;
6. import java.io.IOException;
7. import java.util.Scanner;
8. import java.util.regex.Matcher;
9. import java.util.regex.Pattern;
10.
11. public class GrepCommandImplementation
12. {
13.     public static void main(String[] args) throws Exception
14.     {
15.         Scanner sc = new Scanner(System.in);
16.         System.out
17.             .println("Enter the string to match from GrepCommandImplementation.java file: ");
18.         Pattern pattern = Pattern.compile(sc.next());
19.         Matcher matcher = pattern.matcher("");
20.         String file = "src/com/sanfoundry/setandstring/GrepCommandImplementation.java";
21.         BufferedReader br = null;
22.         String line;
23.         try
24.         {
25.             br = new BufferedReader(new FileReader(file));
26.         }
27.         catch (IOException e)
28.         {
29.             System.err.println("Cannot read " + file + ": " + e.getMessage());
30.         }
31.         while ((line = br.readLine()) != null)
32.         {
33.             matcher.reset(line);
34.             if (matcher.find())
35.             {
36.                 System.out.println(file + ": " + line);
37.             }
38.         }
39.         br.close();
40.         sc.close();
41.     }
42. }
```

Output:

advertisement

```
$ javac GrepCommandImplementation.java
$ java GrepCommandImplementation
```

```
Enter the string to match from GrepCommandImplementation.java file:
println
```

```
src/com/sanfoundry/setandstring/GrepCommandImplementation.java:  
GrepCommandImplementation.java file: ");  
src/com/sanfoundry/setandstring/GrepCommandImplementation.java:  
e.getMessage());  
src/com/sanfoundry/setandstring/GrepCommandImplementation.java:  
.println("Enter the string to match from  
System.err.println("Cannot read '" + file + "': " +  
System.out.println(file +
```

364.Java Program to Implement Aho-Corasick Algorithm for String Matching

[« Prev](#)

[Next »](#)

This is a java program to implement Aho-Corasick Algorithm. It is a kind of dictionary-matching algorithm that locates elements of a finite set of strings (the “dictionary”) within an input text. It matches all patterns simultaneously. The complexity of the algorithm is linear in the length of the patterns plus the length of the searched text plus the number of output matches. Note that because all matches are found, there can be a quadratic number of matches if every substring matches (e.g. dictionary = a, aa, aaa, aaaa and input string is aaaa).

Here is the source code of the Java Program to Implement Aho-Corasick Algorithm for String Matching. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Arrays;
5. import java.util.Scanner;
6.
7. public class StringSearchUsingAhoCorasickAlgo
8. {
9.     static final int ALPHABET_SIZE = 26;
10.    Node[] nodes;
11.    int nodeCount;
12.
13.    public static class Node
14.    {
15.        int parent;
16.        char charFromParent;
17.        int suffLink = -1;
18.        int[] children = new int[ALPHABET_SIZE];
19.        int[] transitions = new int[ALPHABET_SIZE];
20.        boolean leaf;
21.
22.        {
23.            Arrays.fill(children, -1);
24.            Arrays.fill(transitions, -1);
25.        }
26.
27.    public StringSearchUsingAhoCorasickAlgo(int maxNodes)
28.    {
29.        nodes = new Node[maxNodes];
30.        // create root
31.        nodes[0] = new Node();
32.        nodes[0].suffLink = 0;
33.        nodes[0].parent = -1;
34.        nodeCount = 1;
35.    }
36.
37.    public void addString(String s)
38.    {
39.        int cur = 0;
40.        for (char ch : s.toCharArray())
41.        {
42.            int c = ch - 'a';
43.            if (nodes[cur].children[c] == -1)
44.            {
45.                nodes[nodeCount] = new Node();
46.                nodes[nodeCount].parent = cur;
47.                nodes[nodeCount].charFromParent = ch;
48.                nodes[cur].children[c] = nodeCount++;
49.            }
50.        }
51.    }
52.
53.    public void search(String s)
54.    {
55.        int cur = 0;
56.        for (int i = 0; i < s.length(); i++)
57.        {
58.            int c = s.charAt(i) - 'a';
59.            if (nodes[cur].children[c] == -1)
60.            {
61.                System.out.println("Pattern not found");
62.                return;
63.            }
64.            cur = nodes[cur].children[c];
65.        }
66.        System.out.println("Pattern found at index " + i);
67.    }
68.
69.    public static void main(String[] args)
70.    {
71.        StringSearchUsingAhoCorasickAlgo algo = new StringSearchUsingAhoCorasickAlgo();
72.        algo.addString("aa");
73.        algo.addString("aaa");
74.        algo.addString("aaaa");
75.        algo.addString("a");
76.        algo.search("aaaa");
77.    }
78.}
```

```

49.     }
50.     cur = nodes[cur].children[c];
51.   }
52.   nodes[cur].leaf = true;
53. }
54.
55. public int suffLink(int nodeIndex)
56. {
57.   Node node = nodes[nodeIndex];
58.   if (node.suffLink == -1)
59.     node.suffLink = node.parent == 0 ? 0 : transition(
60.       suffLink(node.parent), node.charFromParent);
61.   return node.suffLink;
62. }
63.
64. public int transition(int nodeIndex, char ch)
65. {
66.   int c = ch - 'a';
67.   Node node = nodes[nodeIndex];
68.   if (node.transitions[c] == -1)
69.     node.transitions[c] = node.children[c] != -1 ? node.children[c]
70.       : (nodeIndex == 0 ? 0 : transition(suffLink(nodeIndex), ch));
71.   return node.transitions[c];
72. }
73.
74. // Usage example
75. public static void main(String[] args)
76. {
77.   StringSearchUsingAhoCorasickAlgo ahoCorasick = new StringSearchUsingAhoCorasickAlgo(
78.     1000);
79.   Scanner sc = new Scanner(System.in);
80.   System.out.println("Enter the main string: ");
81.   String main = sc.nextLine().toLowerCase().trim();
82.   System.out.println("Enter the pattern string: ");
83.   String pattern = sc.nextLine().toLowerCase().trim();
84.   ahoCorasick.addString(pattern);
85.   int node = 0;
86.   for (char ch : main.toCharArray())
87.   {
88.     node = ahoCorasick.transition(node, ch);
89.   }
90.   if (ahoCorasick.nodes[node].leaf)
91.     System.out.println("A " + pattern + " string is substring of "
92.       + main + " string.");
93.   else
94.     System.out.println("A " + pattern
95.       + " string is not substring of " + main + " string.");
96.   sc.close();
97. }
98.

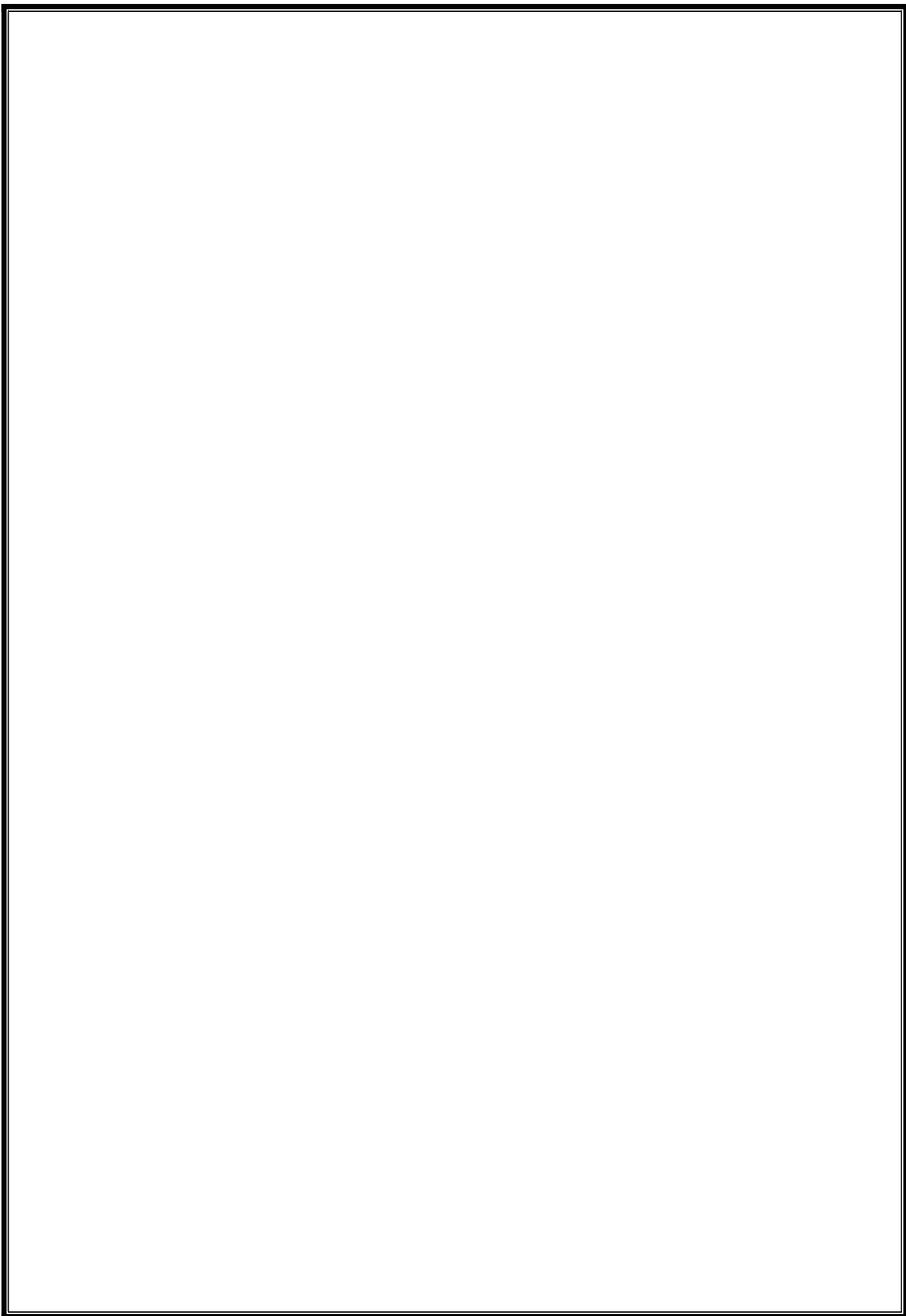
```

Output:

advertisement

```
$ javac StringSearchUsingAhoCorasickAlgo.java
$ java StringSearchUsingAhoCorasickAlgo
```

```
Enter the main string:
Sanfoundry
Enter the pattern string:
Asanfoundry
A 'asanfoundry' string is not substring of sanfoundry string.
```



365.Java Program to Repeatedly Search the Same Text (such as Bible by building a Data Structure)

[« Prev](#)

[Next »](#)

This is a java program to repeatedly search a string from the sample string.

Here is the source code of the Java Program to Repeatedly Search the Same Text (such as Bible by building a Data Structure). The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. class BoyerMooreAlgo
7. {
8.     private final int BASE;
9.     private int[] occurrence;
10.    private String pattern;
11.
12.    public BoyerMooreAlgo(String pattern)
13.    {
14.        this.BASE = 256;
15.        this.pattern = pattern;
16.        occurrence = new int[BASE];
17.        for (int c = 0; c < BASE; c++)
18.            occurrence[c] = -1;
19.        for (int j = 0; j < pattern.length(); j++)
20.            occurrence[pattern.charAt(j)] = j;
21.    }
22.
23.    public void search(String text)
24.    {
25.        int n = text.length();
26.        int m = pattern.length();
27.        int skip = 0;
28.        for (int i = 0; i < (n - m); i += skip)
29.        {
30.            skip = 0;
31.            for (int j = m - 1; j >= 0; j--)
32.            {
33.                if (pattern.charAt(j) != text.charAt(i + j))
34.                {
35.                    skip = Math.max(1, j - occurrence[text.charAt(i + j)]);
36.                    break;
37.                }
38.            }
39.            if (skip == 0)
40.            {
41.                System.out.println("The text " + pattern
42.                    + " is first found after the " + i + " position.");
43.                i++;
44.            }
45.        }
46.        if (skip == 0)
47.            System.out.println("The text " + pattern
48.                + " is first found after the " + n + " position.");
49.    }
50.}
51.
52. public class ReperatedStringSearch
```

```
53. {
54.     public static void main(String[] args)
55.     {
56.         Scanner sc = new Scanner(System.in);
57.         System.out.print("Enter the main string: ");
58.         String text = sc.nextLine();
59.         System.out.print("Enter the string to be searched: ");
60.         String pattern = sc.nextLine();
61.         BoyerMooreAlgo bm = new BoyerMooreAlgo(pattern);
62.         bm.search(text);
63.         sc.close();
64.     }
65. }
```

Output:

advertisement

```
$ javac RepeatedStringSearch.java
$ java RepeatedStringSearch
```

Enter the main string: Sanfoundry is No. 1 choice for Deep Hands-ON Trainings in SAN, Linux & C, Kernel & Device Driver Programming. Our Founder has trained employees of almost all Top Companies in India. Here are few of them: VMware, Citrix, Oracle, Motorola, Ericsson, Aricent, HP, Intuit, Microsoft, Cisco, SAP Labs, Siemens, Symantec, Redhat, Chelsio, Cavium Networks, ST Microelectronics, Samsung, LG-Soft, Wipro, TCS, HCL, IBM, Accenture, HSBC, Northwest Bank, Mphasis, Tata Elxsi, Tata Communications, Mindtree, Cognizant, mid size IT companies and many Startups. Students from top Universities and colleges such as NIT Trichy, BITS Pilani, University of California, Irvine, University of Texas, Austin & PESIT Bangalore have benefited a lot from these courses as well. The assignments and real time projects for our courses are of extremely high quality with excellent learning curve.

Enter the string to be searched: C

The text 'C' is first found after the 71 position.
The text 'C' is first found after the 162 position.
The text 'C' is first found after the 212 position.
The text 'C' is first found after the 280 position.
The text 'C' is first found after the 324 position.
The text 'C' is first found after the 333 position.
The text 'C' is first found after the 397 position.
The text 'C' is first found after the 402 position.
The text 'C' is first found after the 425 position.
The text 'C' is first found after the 470 position.
The text 'C' is first found after the 496 position.
The text 'C' is first found after the 639 position.

366.Java Program to Implement the String Search Algorithm for Short Text Sizes

[« Prev](#)

[Next »](#)

This is a java program to search string using simple algorithm BoyerMoore.

Here is the source code of the Java Program to Implement the String Search Algorithm for Short Text Sizes. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.Scanner;
5.
6. class BoyerMoore
7. {
8.     private final int BASE;
9.     private int[] occurrence;
10.    private String pattern;
11.
12.    public BoyerMoore(String pattern)
13.    {
14.        this.BASE = 256;
15.        this.pattern = pattern;
16.        occurrence = new int[BASE];
17.        for (int c = 0; c < BASE; c++)
18.            occurrence[c] = -1;
19.        for (int j = 0; j < pattern.length(); j++)
20.            occurrence[pattern.charAt(j)] = j;
21.    }
22.
23.    public int search(String text)
24.    {
25.        int n = text.length();
26.        int m = pattern.length();
27.        int skip;
28.        for (int i = 0; i <= n - m; i += skip)
29.        {
30.            skip = 0;
31.            for (int j = m - 1; j >= 0; j--)
32.            {
33.                if (pattern.charAt(j) != text.charAt(i + j))
34.                {
35.                    skip = Math.max(1, j - occurrence[text.charAt(i + j)]);
36.                    break;
37.                }
38.            }
39.            if (skip == 0)
40.                return i;
41.        }
42.        return n;
43.    }
44.}
45.
46. public class StringSearch
47.{
48.    public static void main(String[] args)
49.    {
50.        Scanner sc = new Scanner(System.in);
51.        System.out.print("Enter the main string: ");
52.        String text = sc.nextLine();
53.        System.out.print("Enter the string to be searched: ");
```

```
54. String pattern = sc.nextLine();
55. BoyerMoore bm = new BoyerMoore(pattern);
56. int first_occur_position = bm.search(text);
57. System.out.println("The text " + pattern
58.         + " is first found after the " + first_occur_position
59.         + " position.");
60. sc.close();
61. }
62.}
```

Output:

advertisement

```
$ javac StringSearch.java
$ java StringSearch
```

Enter the main string: I will soon be working in Redmond USA with Mircosoft.

Enter the string to be searched: soon

The text 'soon' is first found after the 7 position.

367.Java Program to Solve Set Cover Problem assuming at max 2 Elements in a Subset

[« Prev](#)

This is a java program to solve set cover problem. The set covering problem (SCP) is a classical question in combinatorics, computer science and complexity theory. Given a set of elements $\{1, 2, \dots, m\}$ (called the universe) and a set S of n sets whose union equals the universe, the set cover problem is to identify the smallest subset of S whose union equals the universe. For example, consider the universe $U = \{1, 2, 3, 4, 5\}$ and the set of sets $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. Clearly the union of S is U . However, we can cover all of the elements with the following, smaller number of sets: $\{\{1, 2, 3\}, \{4, 5\}\}$.

Here is the source code of the Java Program to Solve Set Cover Problem assuming at max 2 Elements in a Subset. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```
1.
2. package com.sanfoundry.setandstring;
3.
4. import java.util.ArrayList;
5. import java.util.Arrays;
6. import java.util.Collections;
7. import java.util.Comparator;
8. import java.util.LinkedHashSet;
9. import java.util.List;
10.import java.util.Set;
11.
12.public class SetCoverMax2Elem
13.
14. interface Filter<T>
15. {
16.     boolean matches(T t);
17. }
18.
19. private static <T> Set<T> shortestCombination(Filter<Set<T>> filter,
20.         List<T> listOfSets)
21. {
22.     final int size = listOfSets.size();
23.     if (size > 20)
24.         throw new IllegalArgumentException("Too many combinations");
25.     int combinations = 1 << size;
26.     List<Set<T>> possibleSolutions = new ArrayList<Set<T>>();
27.     for (int l = 0; l < combinations; l++)
28.     {
29.         Set<T> combination = new LinkedHashSet<T>();
30.         for (int j = 0; j < size; j++)
31.         {
32.             if (((l >> j) & 1) != 0)
33.                 combination.add(listOfSets.get(j));
34.         }
35.         possibleSolutions.add(combination);
36.     }
37. // the possible solutions in order of size.
38. Collections.sort(possibleSolutions, new Comparator<Set<T>>()
39. {
40.     public int compare(Set<T> o1, Set<T> o2)
41.     {
42.         return o1.size() - o2.size();
43.     }
44. });
45. for (Set<T> possibleSolution : possibleSolutions)
46. {
47.     if (filter.matches(possibleSolution))
```

```

48.         return possibleSolution;
49.     }
50.     return null;
51. }
52.
53. public static void main(String[] args)
54. {
55.     Integer[][] arrayOfSets = { { 1, 2 }, { 3, 8 }, { 9, 10 }, { 1, 10 },
56.         { 2, 3 }, { 4, 5 }, { 5, 7 }, { 5, 6 }, { 4, 7 }, { 6, 7 },
57.         { 8, 9 }, };
58.     Integer[] solution = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
59.     List<Set<Integer>> listOfSets = new ArrayList<Set<Integer>>();
60.     for (Integer[] array : arrayOfSets)
61.         listOfSets.add(new LinkedHashSet<Integer>(Arrays.asList(array)));
62.     final Set<Integer> solutionSet = new LinkedHashSet<Integer>(
63.         Arrays.asList(solution));
64.     Filter<Set<Set<Integer>>> filter = new Filter<Set<Set<Integer>>>()
65.     {
66.         public boolean matches(Set<Set<Integer>> integers)
67.         {
68.             Set<Integer> union = new LinkedHashSet<Integer>();
69.             for (Set<Integer> ints : integers)
70.                 union.addAll(ints);
71.             return union.equals(solutionSet);
72.         }
73.     };
74.     Set<Set<Integer>> firstSolution = shortestCombination(filter,
75.         listOfSets);
76.     System.out.println("The shortest combination was " + firstSolution);
77. }
78.

```

Output:

advertisement

```
$ javac SetCoverMax2Elem.java
$ java SetCoverMax2Elem
```

The shortest combination was [[1, 2], [3, 8], [9, 10], [5, 6], [4, 7]]

