

**UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**MAHRSHI DAYANAD UNIVERSITY**

**ROHTAK**

**PROGRAMMING IN JAVA**

**Assignment-1**

**Submitted by :-**

Name - Himanshu Kumar

Roll no – 27650

Branch – CSE-A

Semester - 5th Sem

Course code - PCC-CSE-309G

# Submitted to :-

Dr. Dheeraj sahni

# 

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Assignment – 1 java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Java Basic Programs

1. Fibonacci Series in Java

The Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. In other words, it is an infinite sequence of integers where each number is the sum of the two numbers immediately before it. The Fibonacci sequence starts like this:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

**PROGRAM :-**

public class assignment {

public static void main(String[] args) {

int n = 10;

System.out.println(fibonacci(+n));

System.out.println("Fibonacci Series:");

for (int i = 0; i <= n; i++) {

System.out.print(fibonacci(i) + " ");

}

}

public static long fibonacci(int n) {

if (n <= 1) {

return n;

} else {

return fibonacci(n - 1) + fibonacci(n - 2);

}

}

}

**OUTPUT :-**

55

Fibonacci Series:

0 1 1 2 3 5 8 13 21 34 55

1. Prime Number Program in Java

class prime{

public static void main(String[] args) {

int n=93;

if (n==1 || n == 2 || n == 3) {

System.out.println("the prime number is = "+n);

}

else if(n % 2==0 || n % 3 ==0)

{

System.out.println(n+" is not a prime number");

}

else{

System.out.println("the prime number is = "+n);

}

}

}

**OUTPUT:-** 93 is not a prime number

the prime number is = 97

1. Palindrome Program in Java

**Program :-**

class palindrome{

public static void main(String[] args) {

String str = "abab";

//String str = "madam";

String str2 ="";

for ( int i = str.length()-1; i >= 0; i--) {

str2 = str2+ str.charAt(i);

}

if(str.equals(str2))System.out.println("this is palindrome : "+str2);

else System.out.println(str2+" or "+str+" is not same");

}

}

**OUTPUT:-**

baba or abab is not same

this is palindrome : madam

1. Factorial Program in Java

Program:-

class Factorial{

public static void main(String[] args) {

int number = 5;

int answer=1;

for(int i = number; i > 0; i--){

answer= answer\*i;

}

System.out.print("the factorial of "+number+" is "+answer);

}

}

Output :- the factorial of 5 is 120

1. Armstrong Number in Java

class armstrong {

public static void main(String[] args) {

int armstrongNumber = 153;

int digits, result=0;

int originalNumber = armstrongNumber;

int power = 0;

while (armstrongNumber != 0) {

armstrongNumber/=10;

power++;

}

armstrongNumber= originalNumber;

while (armstrongNumber != 0) {

digits = armstrongNumber % 10;

result += Math.pow(digits, power);

armstrongNumber /= 10;

} if (result == originalNumber) {

System.out.println(originalNumber + " is an Armstrong number.");

} else {

System.out.println(originalNumber + " is not an Armstrong number.");

}

}

}

Opuput:-

153 is an Armstrong number.

1. How to Generate Random Numbers in Java

class Rendome{

public static void main(String[] args) {

System.out.println("1st randome number "+Math.random());

System.out.println("2nd randome number "+Math.random());

}

}

1st randome number 0.944565041163729

2nd randome number 0.8359087168193496

1. How to Print Pattern in Java

class patten{

public static void main(String[] args) {

int size=5;

// pattern(size);

int n=5;

for (int row=1; row<n; row++){

for(int col=1; col<=row; col++){

System.out.print("\* ");

}

System.out.println(" ");

}

}

}

\*

\* \*

\* \* \*

\* \* \* \*

1. How to Compare Two Objects in Java

class CompareTwoOBject{

public static void main(String[] args) {

String name = new String("Himanshu");// creating object of name variable

String name2 = new String("Himanshu");// creating object of name2 variable

String name3 = new String("himanshu");// creating object of name3 variable

// comparing objects

System.out.println("is "+name+" is equal to "+name2+" "+name.equals(name2));

System.out.println("is "+name+" is equal to "+name3+" "+name.equals(name3));

}

}

is Himanshu is equal to Himanshu true

is Himanshu is equal to himanshu false

1. How to Create Object in Java

class Calculator{

int a,b,c;

int add(int a, int b){

return a+b;

}float add(int a, float b,int c){

return a+b+c;

}double add(double a, int b){

return a+b;

}double add(double a, double b){

return a+b;

}float add(float a, float b){

return a+b;

}

}

public class constractor {

public static void main(String[] args) {

Calculator cal = new Calculator(); //creating objects of calculator class

System.out.println("addition of two number "+cal.add(10,20)); // passing argument in add function

}

1. How to Print ASCII Value in Java

class Ascii {

public static void main(String[] args) {

char a = 'Z';

System.out.println("ASCII code of Z is " +Character.getNumericValue(a));

}

}

ASCII code of Z is 35

Java Array Programs

1. Java Program to copy all elements of one array into another array

class copyElements{

public static void main(String[] args) {

int [] arr ={10,5,6,55,8,5,9,5};

int[] secondArray = new int[arr.length];

for (int i = 0 , j= 0; i < arr.length; i++) {

secondArray[j] = arr[i];

j++;

}

System.out.println(Arrays.toString(secondArray));

}

}

[10, 5, 6, 55, 8, 5, 9, 5]

1. Java Program to find the frequency of each element in the array

class FindFrequency{

public static void main(String[] args) {

int [ ]arr = {1,5,6,3,2,5,1,4,1};

// Arrays.sort(arr);

int FindFrequencyOf = 1;

System.out.println(Arrays.toString(arr));

System.out.println("Frequency of "+FindFrequencyOf+" is "+Times(arr, FindFrequencyOf)); }

public static int Times(int[] arr,int value){

int times= 0;

for (int i = 0,j=0; i < arr.length; i++) {

if (arr[i] == value) {

times++;

}

}

return times;

}

}

[1, 5, 6, 3, 2, 5, 1, 4, 1]

Frequency of 1 is 3

1. Java Program to left rotate the elements of an array

class LeftRotate{

public static void main(String[] args) {

int[] arr = {5,6,9,7,4};

int rotate=1;

int[] rotateArray= new int[arr.length];

for(int i = 0; i< arr.length; i++){

int newIndex = (i - rotate + arr.length)% arr.length;

rotateArray[newIndex] = arr[i];

}

System.out.println(Arrays.toString(arr));

System.out.println(Arrays.toString(rotateArray));

}

}

Befor :- [5, 6, 9, 7, 4]

After :- [6, 9, 7, 4, 5]

4) Java Program to print the duplicate elements of an array

5) Java Program to print the elements of an array

class PrintElementOfArrays{

public static void main(String[] args) {

int[] arr = {1,5,6,8,4,2,3,4};

System.out.println("printing array elements :- ");

for (int i=0; i<arr.length; i++) {

System.out.print(" "+arr[i]);

}

}

}

printing array elements :-

1 5 6 8 4 2 3 4

6) Java Program to print the elements of an array in reverse order

class PrintElementOfArraysInReverse{

public static void main(String[] args) {

int[] arr = {1,5,6,8,4,2,3,4};

System.out.println("printing array elements in reverse order :- ");

for (int i=arr.length-1; i>=0; i--) {

System.out.print(" "+arr[i]);

}

}

}

printing array elements in reverse order :-

4 3 2 4 8 6 5 1

7) Java Program to print the elements of an array present on even position

class PrintElementOfArraysATEvenPlace{

public static void main(String[] args) {

int[] arr = {1,5,6,8,4,2,3,4};

System.out.println("printing array elements At even place :- ");

for (int i=0; i<arr.length; i=+2) {

System.out.print(" "+arr[i]);

}

}

}

printing array elements At even place :-

1 6 4 3

8) Java Program to print the elements of an array present on odd position

class PrintElementOfArraysAtOddPlace{

public static void main(String[] args) {

int[] arr = {1,5,6,8,4,2,3,4};

System.out.println("printing array elements At Odd place :- ");

for (int i=1; i<=arr.length; i+=2) {

System.out.print(" "+arr[i]);

}

}

}

printing array elements At Odd place :-

5 8 2 4

9) Java Program to print the largest element in an array

class LargestElemnt{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

int largeValue = 0;

for (int i = 0; i < arr.length; i++) {

if (arr[i] > largeValue ) {

largeValue = arr[i];

}

}

System.out.println("largest value among all is "+largeValue);

}

}

largest value among all is 8

1. Java Program to print the smallest element in an array

class SmallestElement{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

int smallestValue = arr[0];

for (int i = 0; i < arr.length; i++) {

if (arr[i] < smallestValue ) {

smallestValue = arr[i];

}

}

System.out.println("largest value among all is "+smallestValue);

}

}

largest value among all is 2

1. Java Program to print the number of elements present in an array

class NumberOfElement{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

System.out.println("Total number of element:- "+arr.length);

}

}

Total number of element:-8

1. Java Program to print the sum of all the items of the array

class AddOfAllElement{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

int sum=0;

for (int i = 0; i < arr.length; i++) {

sum = sum + arr[i];

}

System.out.println("sum number of element:- "+sum);

}

}

sum number of element:- 37

1. Java Program to right rotate the elements of an array

class RightRotate{

public static void main(String[] args) {

int[] arr = {5,6,9,7,4};

int rotate=1;

int[] rotateArray= new int[arr.length];

for(int i = 0; i< arr.length; i++){

int newIndex = (i + rotate + arr.length)% arr.length;

rotateArray[newIndex] = arr[i];

}

System.out.println("befor Rotate"+Arrays.toString(arr));

System.out.println("After Rotate"+Arrays.toString(rotateArray));

}

}

befor Rotate[5, 6, 9, 7, 4]

After Rotate[4, 5, 6, 9, 7]

1. Java Program to sort the elements of an array in ascending order

class AssendingOrder{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

Arrays.sort(arr);

System.out.println("Array in increasing order :- "+Arrays.toString(arr));

}

} Array in increasing order :- [2, 3, 3, 4, 4, 6, 7, 8]

1. Java Program to sort the elements of an array in descending order

class AssendingOrder{

public static void main(String[] args) {

Integer[] arr = {7,3,6,8,4,2,3,4};

Arrays.sort(arr);

List<Integer> list = new ArrayList<>(Arrays.asList(arr));

Collections.reverse(list);

Integer[] reversedArray = list.toArray(new Integer[0]);

System.out.println("Original Array: " + Arrays.toString(arr));

System.out.println("Reversed Array: " + Arrays.toString(reversedArray));

}

} Original Array: [2, 3, 3, 4, 4, 6, 7, 8]

Reversed Array: [8, 7, 6, 4, 4, 3, 3, 2]

1. Java Program to Find 3rd Largest Number in an array

class rdLargestElemnt{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

Arrays.sort(arr);

System.out.println(Arrays.toString(arr));

System.out.println("3rd largest number among all is "+arr[arr.length-3]);

}

}

[2, 3, 3, 4, 4, 6, 7, 8]

3rd largest number among all is 6

1. Java Program to Find 2nd Largest Number in an array

class rdLargestElemnt{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

Arrays.sort(arr);

System.out.println(Arrays.toString(arr));

System.out.println("3rd largest number among all is "+arr[arr.length-2]);

}

}

[2, 3, 3, 4, 4, 6, 7, 8]

2rd largest number among all is 7

1. Java Program to Find Largest Number in an array

class LargestElemnt{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

int largeValue = 0;

for (int i = 0; i < arr.length; i++) {

if (arr[i] > largeValue ) {

largeValue = arr[i];

}

}

System.out.println("largest value among all is "+largeValue);

}

}

largest value among all is 8

1. Java to Program Find 2nd Smallest Number in an array

class ndSmallest{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

Arrays.sort(arr);

System.out.println(Arrays.toString(arr));

System.out.println("2rd smallest number among all is "+arr[1]);

}

} [2, 3, 3, 4, 4, 6, 7, 8]

2rd smallest number among all is 3

1. Java Program to Find Smallest Number in an array

class SmallestElement{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

int smallestValue = arr[0];

for (int i = 0; i < arr.length; i++) {

if (arr[i] < smallestValue ) {

smallestValue = arr[i];

}

}

System.out.println("largest value among all is "+smallestValue);

}

}

largest value among all is 2

1. Java Program to Remove Duplicate Element in an array
2. Java Program to Print Odd and Even Numbers from an array

class EvenOrOdd{

public static void main(String[] args) {

int[] arr = {7,3,10,15,62,5,45,89,6,8,4,2,4};

Arrays.sort(arr);

int[] EvenNumber =Even(arr);

int[] OddNumber =Odd(arr);

System.out.println("Even number of the array : "+Arrays.toString(EvenNumber));

System.out.println("odd number of the array : "+Arrays.toString(OddNumber));

}

public static int[] Even(int[]array){

int[] even = new int[array.length];

int j = 0;

for (int i = 0 ; i < array.length; i++) {

if (array[i]%2==0) {

even[j] =array[i];

j++;

}

}

return Arrays.copyOf(even, j);

}

public static int[] Odd(int[]array){

int[] odd = new int[array.length];

int j = 0;

for (int i = 0 ; i < array.length; i++) {

if (array[i]%2 != 0) {

odd[j] = array[i];

j++;

}

}

return Arrays.copyOf(odd, j);

}

}

Even number of the array : [2, 4, 4, 6, 8, 10, 62]

odd number of the array : [3, 5, 7, 15, 45, 89]

3.

1. How to Sort an Array in Java

In Java, you can sort an array using either the built-in sorting methods or by implementing your own sorting algorithm.

class Sort{

public static void main(String[] args) {

int[] arr = {7,3,6,8,4,2,3,4};

System.out.println("Befor sort "+Arrays.toString(arr));

Arrays.sort(arr);

System.out.println("After sort "+Arrays.toString(arr));

}

}

Befor sort [7, 3, 6, 8, 4, 2, 3, 4]

After sort [2, 3, 3, 4, 4, 6, 7, 8]

Java Matrix Programs

1. Java Matrix Programs
2. Java Program to Add Two Matrices

class AddTwoMatrix{

public static void main(String[] args) {

int[][] arr = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] arr2 = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] result = new int[arr.length][arr[0].length];

// int [][] result = new int [arr.length][];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

result[i][j] = arr[i][j] + arr2[i][j];

}

}

System.out.println("after the addition of Add Two Matrices");

for (int i = 0; i < result.length; i++) {

for (int j = 0; j < result.length; j++) {

System.out.print(" "+result[i][j]);

}

System.out.println(" ");

}

}

}

after the addition of Add Two Matrices

24 10 12 6

2 130 18 16

8 12 10 8

30 8 12 42

1. Java Program to Multiply Two Matrices

class MulTwoMatrix{

public static void main(String[] args) {

int[][] arr = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] arr2 = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] result = new int[arr.length][arr[0].length];

// int [][] result = new int [arr.length][];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

result[i][j] = arr[i][j] \* arr2[i][j];

}

}

System.out.println("after the multiplication of Two Matrices");

for (int i = 0; i < result.length; i++) {

for (int j = 0; j < result.length; j++) {

System.out.print(" "+result[i][j]);

}

System.out.println(" ");

}

}

}

after the multiplication of Two Matrices

144 25 36 9

1 4225 81 64

16 36 25 16

225 16 36 441

1. Java Program to subtract the two matrices

class SubTwoMatrix{

public static void main(String[] args) {

int[][] arr = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] arr2 = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] result = new int[arr.length][arr[0].length];

// int [][] result = new int [arr.length][];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

result[i][j] = arr[i][j] - arr2[i][j];

}

}

System.out.println("after the substraction of Two Matrices");

for (int i = 0; i < result.length; i++) {

for (int j = 0; j < result.length; j++) {

System.out.print(" "+result[i][j]);

}

System.out.println(" ");

}

}

}

after the substraction of Two Matrices

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

1. Java Program to determine whether two matrices are equal

class TwoMatrixisequl{

public static void main(String[] args) {

int[][] arr = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] arr2 = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

if (twoMatrixeql(arr, arr2)) {

System.out.println("Matrices are equal");

}

else System.out.println("matrices are not equal");

}

public static Boolean twoMatrixeql(int[][] arr, int[][] arr2){

if (arr.length != arr2.length || arr[0].length != arr2[0].length) {

return false;

}

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

if(arr[i][j] != arr2[i][j]){

return false;

}

}

}

return true;

}

}matrices are equal

1. Java Program to display the lower triangular matrix

class LowerTriangleMatrix{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,6,9}

};

int [][]newMatrix =new int[arr.length][arr.length];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j <= i; j++) {

newMatrix [i][j] = arr[i][j];

}

}

System.out.println("Lower triangle of the Matrices");

for (int i = 0; i < newMatrix.length; i++) {

for (int j = 0; j < newMatrix.length; j++) {

System.out.print(" "+newMatrix[i][j]);

}

System.out.println(" ");

}

}

}

Lower triangle of the Matrices

2 0 0 0

1 5 0 0

4 6 5 0

1 4 6 9

1. Java Program to display the upper triangular matrix

class UpperTriangleMatrix{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,6,9}

};

int [][]newMatrix =new int[arr.length][arr.length];

for (int i = 0; i < arr.length; i++) {

for (int j = arr.length-1; j >= i; j--) {

newMatrix [i][j] = arr[i][j];

}

}

System.out.println("Upper triangle of the Matrices");

for (int i = 0; i < newMatrix.length; i++) {

for (int j = 0; j < newMatrix.length; j++) {

System.out.print(" "+newMatrix[i][j]);

}

System.out.println(" ");

}

}

}

Upper triangle of the Matrices

2 5 6 3

0 5 9 8

0 0 5 4

0 0 0 9

1. Java Program to find the frequency of odd & even numbers in the given matrix

class FindFrequencyMatrix{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,3,9}

};

System.out.println("total number of even frequency : "+even(arr));

System.out.println("total number of odd frequency : "+Odd(arr));

}

public static int even ( int[][]arr ){

int Even =0;

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

if( arr[i][j] %2 == 0){

Even++;

}

}

}

return Even;

}

public static int Odd ( int[][]arr ){

int odd =0;

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

if( arr[i][j] %2 != 0){

odd++;

}

}

}

return odd;

}

}

total number of even frequency : 7

total number of odd frequency : 9

9) Java Program to find the product of two matrices

class MulTwoMatrix{

public static void main(String[] args) {

int[][] arr = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] arr2 = {

{12,5,6,3},

{1,65,9,8},

{4,6,5,4,1},

{15,4,6,21}

};

int[][] result = new int[arr.length][arr[0].length];

// int [][] result = new int [arr.length][];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

result[i][j] = arr[i][j] \* arr2[i][j];

}

}

System.out.println("after the multiplication of Two Matrices");

for (int i = 0; i < result.length; i++) {

for (int j = 0; j < result.length; j++) {

System.out.print(" "+result[i][j]);

}

System.out.println(" ");

}

}

}

after the multiplication of Two Matrices

144 25 36 9

1 4225 81 64

16 36 25 16

225 16 36 441

1. Java Program to find the sum of each row and each column of a matrix

class SumOfrowAndColumn{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,3,9}

};

int [][] sumOfcolumn = new int [arr.length][arr[0].length];

for (int i = 0; i < arr.length; i++) {

int sum =0;

for (int j = 0; j < arr[0].length; j++) {

sumOfcolumn[i][j] += arr[i][j];

sum += arr[i][j];

}

System.out.println("sum of "+i+" row "+sum);

}

System.out.println("column");

for (int i = 0; i < arr.length; i++) {

int sum =0;

for (int j = 0; j < arr[0].length; j++) {

sumOfcolumn[i][j] += arr[i][j];

sum += arr[j][i];

}

System.out.println("sum of "+i+" column "+sum);

}

}

}

sum of 0 row 16

sum of 1 row 23

sum of 2 row 19

sum of 3 row 17

column

sum of 0 column 8

sum of 1 column 20

sum of 2 column 23

sum of 3 column 24

1. Java Program to find the transpose of a given matrix

class TransposeMatrix{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,3,9}

};

int [][] newMatrix = new int [arr.length][arr[0].length];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

newMatrix[i][j] = arr[j][i];

}

}

for (int i = 0; i < newMatrix.length; i++) {

for (int j = 0; j < newMatrix.length; j++) {

System.out.print(" "+ newMatrix[i][j]);

}

System.out.println();

}

}}

'TransposeMatrix'

2 1 4 1

5 5 6 4

6 9 5 3

3 8 4 9

1. Java Program to determine whether a given matrix is an identity matrix

class IdentityMatrix{

public static void main(String[] args) {

int [][]arr = {

{1, 0, 0},

{0, 1, 0},

{0, 0, 1}

} ;

if (checkIdendity(arr)) {

System.out.println("matrix is identity");

}else

System.out.println(" matrix is not identity");

}

public static Boolean checkIdendity(int[][]arr){

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

if (i==j && arr[i][j] !=1) {

return false;

}

else if (i != j && arr[i][j]!= 0) {

return false;

}

}

}

return true;

}

}

Matrix is identity

1. Java Program to determine whether a given matrix is a sparse matrix

class sparseMatrix{

public static void main(String[] args) {

int [][]arr = {

{1, 0, 0},

{0, 1, 0},

{0, 0, 1}

} ;

int hold = 0;

Boolean ismatrix = sparse(arr,hold);

if (ismatrix) {

System.out.println("The given matrix is sparse matrix");

}else System.out.println("The given matrix is not sparse matrix");

}

public static Boolean sparse(int arr[][], int hold){

int count = 0;

for ( int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

if (arr[i][j]==0) {

count++;

}

}

}

return count > hold;

}

}

The given matrix is sparse matrix

1. Java Program to transpose matrix

class TransposeMatrix{

public static void main(String[] args) {

int[][] arr = {

{2,5,6,3},

{1,5,9,8},

{4,6,5,4,},

{1,4,3,9}

};

int [][] newMatrix = new int [arr.length][arr[0].length];

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[0].length; j++) {

newMatrix[i][j] = arr[j][i];

}

}

for (int i = 0; i < newMatrix.length; i++) {

for (int j = 0; j < newMatrix.length; j++) {

System.out.print(" "+ newMatrix[i][j]);

}

System.out.println();

}

}}

2 1 4 1

5 5 6 4

6 9 5 3

3 8 4 9

Java String Programs

1. Java Program to count the total number of characters in a string

class TotalChar{

public static void main(String[] args) {

String s1="himanshu";

System.out.println("Total number of characters is "+s1.length());

}

}

Total number of characters is 8

1. Java Program to count the total number of characters in a string 2

class Tchar{

public static void main(String[] args) {

String s2 = "Hello world";

int totalchar= s2.length();

System.out.println("total number of characters is "+totalchar);

}

}

total number of characters is 11

1. Java Program to count the total number of punctuation characters exist in a String

class Pacuaction{

public static void main(String[] args) {

String s1 = "!hi!!!!>???? ?";

Pattern pattern = Pattern.compile("\\p{Punct}");

Matcher matcher = pattern.matcher(s1);

int punch=0;

while (matcher.find()) {

punch++;

}

System.out.println("Total number of punctuation "+punch);

}

}

Total number of punctuation 11

1. Java Program to count the total number of vowels and consonants in a string

class FindVowels{

public static void main(String[] args) {

String name ="Himanshu";

String vowels = "aeiou";

name.toLowerCase();

int totalvol=0;

for (int x = 0; x < name.length(); x++) {

for (int j = 0; j < vowels.length(); j++) {

if (name.charAt(x)==vowels.charAt(j)) {

totalvol++;

}

}

}

System.out.print("Total number of consonants : ");

System.out.println(name.length()-totalvol);

System.out.println("Total number of vowels in char "+totalvol);

}

}

Total number of consonants : 5

Total number of vowels in char 3

1. Java Program to determine whether two strings are the anagram

class anagram{

public static void main(String[] args) {

String str1 = "The classroom";

String str2 = "School Master";

str1=str1.replace(" ", "");

str2= str2.replace(" ", "");

str1=str1.toLowerCase();

str2=str2.toLowerCase();

char arr[]= str1.toCharArray();

char arr1[]= str2.toCharArray();

Arrays.sort(arr);

Arrays.sort(arr1);

if (Arrays.equals(arr, arr1)) {

System.out.println("anagram");

} else {

System.out.println("not anagram");

}

}

}

anagram

1. Java Program to divide a string into 'N' equal parts.

public class DivideStringIntoEqualParts {

public static void main(String[] args) {

String input = "This is a sample string that we want to divide into equal parts.";

int n = 4; // Replace with the desired number of parts (N)

// Calculate the length of each part

int partLength = input.length() / n;

if (partLength == 0) {

System.out.println("Cannot divide the string into equal parts.");

return;

}

System.out.println("Dividing the string into " + n + " equal parts:");

for (int i = 0; i < n; i++) {

int startIndex = i \* partLength;

int endIndex = (i == n - 1) ? input.length() : startIndex + partLength;

String part = input.substring(startIndex, endIndex);

System.out.println("Part " + (i + 1) + ": " + part);

}

}

}

Dividing the string into 4 equal parts:

Part 1: This is a sample

Part 2: string that we

Part 3: want to divide into

Part 4: equal parts.

1. Java Program to find all subsets of a string

public class SubsetsOfString {

public static void main(String[] args) {

String input = "abc";

findSubsets(input);

}

public static void findSubsets(String input) {

findSubsetsRecursive(input, 0, "");

}

public static void findSubsetsRecursive(String input, int index, String currentSubset) {

if (index == input.length()) {

System.out.println(currentSubset); // Print the current subset

return;

}

// Include the current character in the subset

findSubsetsRecursive(input, index + 1, currentSubset + input.charAt(index));

// Exclude the current character from the subset

findSubsetsRecursive(input, index + 1, currentSubset);

}

} abcd

abc

abd

ab

acd

ac

ad

a

bcd

bc

bd

b

cd

c

d

1. Java Program to find the longest repeating sequence in a string

public class LongestRepeatingSequence {

public static void main(String[] args) {

String input = "banana";

String longestRepeatingSequence = findLongestRepeatingSequence(input);

System.out.println("Longest repeating sequence: " + longestRepeatingSequence);

}

public static String findLongestRepeatingSequence(String input) {

int n = input.length();

int[][] dp = new int[n + 1][n + 1];

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (input.charAt(i - 1) == input.charAt(j - 1) && i != j) {

dp[i][j] = dp[i - 1][j - 1] + 1;

} else {

dp[i][j] = Math.max(dp[i][j - 1], dp[i - 1][j]);

}

}

}

int length = dp[n][n];

StringBuilder result = new StringBuilder();

int i = n, j = n;

while (i > 0 && j > 0) {

if (dp[i][j] == dp[i - 1][j - 1] + 1) {

result.insert(0, input.charAt(i - 1));

i--;

j--;

} else if (dp[i][j] == dp[i - 1][j]) {

i--;

} else {

j--;

}

}

return result.toString();

}

}

Longest repeating sequence: ana

1. Java Program to find all the permutations of a string

import java.util.ArrayList;

import java.util.List;

public class StringPermutations {

public static void main(String[] args) {

String input = "abc";

List<String> permutations = findPermutations(input);

System.out.println("Permutations of '" + input + "':");

for (String permutation : permutations) {

System.out.println(permutation);

}

}

public static List<String> findPermutations(String input) {

List<String> result = new ArrayList<>();

permute(input, 0, input.length() - 1, result);

return result;

}

public static void permute(String input, int left, int right, List<String> result) {

if (left == right) {

result.add(input);

} else {

for (int i = left; i <= right; i++) {

input = swap(input, left, i);

permute(input, left + 1, right, result);

input = swap(input, left, i); // backtrack

}

}

}

public static String swap(String input, int i, int j) {

char[] charArray = input.toCharArray();

char temp = charArray[i];

charArray[i] = charArray[j];

charArray[j] = temp;

return new String(charArray);

}

}

Permutations of 'abc':

abc

acb

bac

bca

cab

cba

10) Java Program to remove all the white spaces from a string

class RemoveSpace{

public static void main(String[] args) {

String Name = "Hi This Code remove All Spaces FROM Here";

Name=Name.replace(" ","");

System.out.println(Name);

}

}

HiThisCoderemoveAllSpacesFROMHere

11) Java Program to replace lower-case characters with upper-case and vice-versa

class CaseChanger{

public static void main(String[] args) {

String name = "Hi This Code remove All Spaces FROM Here";

System.out.println("before "+name);

System.out.println("lowercase : "+name.toLowerCase());

System.out.println("uppercase : "+name.toUpperCase())

;

}

} before Hi This Code remove All Spaces FROM Here

lowercase : hi this code remove all spaces from here

uppercase : HI THIS CODE REMOVE ALL SPACES FROM HERE

12) Java Program to replace the spaces of a string with a specific character

class ReplaceSpace{

public static void main(String[] args) {

String Name = "Hi This Code remove All Spaces FROM Here";

System.out.println(Name);

Name=Name.replace(" ","\_");

System.out.println(Name);

}

} Hi This Code remove All Spaces FROM Here

Hi\_This\_Code\_remove\_All\_Spaces\_FROM\_Here

13) Java Program to determine whether a given string is a palindrome

class palindrome{

public static void main(String[] args) {

String str = "abab";

//String str = “madam”;

String str2 ="";

for ( int i = str.length()-1; i >= 0; i--) {

// System.out.println(str.indexOf(i));

str2 = str2+ str.charAt(i);

}

if(str.equals(str2))System.out.println("this is palindrome : "+str2);

else System.out.println(str2+" or "+str+" is not same");

}

}

baba or abab is not same

this is palindrome : madam

14) Java Program to determine whether one string is a rotation of another

15) Java Program to find maximum and minimum occurring characters in a string

16) Java Program to find the Reverse of the string

class reverse{

public static void main(String[] args) {

String str1= "Himanshu kumar";

String str2 ="";

String str[]= str1.split(" ");

for (int i = str.length-1; i >=0; i--) {

str2= str2+str[i]+" ";

}

System.out.println("before reverse : "+str1);

System.out.println("after reverse : "+str2);

}

}

17) Java program to find the duplicate characters in a string

import java.util.HashMap;

import java.util.Map;

public class FindDuplicateCharacters {

public static void main(String[] args) {

String input = "programming";

Map<Character, Integer> charCount = new HashMap<>();

for (char c : input.toCharArray()) {

if (charCount.containsKey(c)) {

charCount.put(c, charCount.get(c) + 1);

} else {

charCount.put(c, 1);

}

}

System.out.println("Duplicate characters in the string '" + input + "':");

for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {

if (entry.getValue() > 1) {

System.out.println(entry.getKey() + " - " + entry.getValue() + " times");

}

}

}

}

Output:-

Duplicate characters in the string 'programming':

r - 2 times

g - 2 times

m - 2 times

18) Java program to find the duplicate words in a string

import java.util.HashMap;

import java.util.Map;

public class FindDuplicateWords {

public static void main(String[] args) {

String input = "This is a test. This is only a test. Testing, one, two, three.";

// Split the input string into words using space as the delimiter

String[] words = input.split(" ");

// Create a HashMap to store word frequencies

Map<String, Integer> wordFrequency = new HashMap<>();

for (String word : words) {

// Remove punctuation and convert to lowercase to standardize words

word = word.replaceAll("[^a-zA-Z]", "").toLowerCase();

if (word.length() > 0) {

if (wordFrequency.containsKey(word)) {

wordFrequency.put(word, wordFrequency.get(word) + 1);

} else {

wordFrequency.put(word, 1);

}

}

}

System.out.println("Duplicate words in the string:");

for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {

if (entry.getValue() > 1) {

System.out.println(entry.getKey() + " - " + entry.getValue() + " times");

}

}

}

}

Output:-

Duplicate words in the string:

a - 2 times

test - 2 times

this - 2 times

is - 2 times

19) Java Program to find the frequency of characters

import java.util.HashMap;

import java.util.Map;

public class CharacterFrequency {

public static void main(String[] args) {

String input = "Hello, World!";

// Remove spaces and convert the string to lowercase for case-insensitive counting

input = input.replaceAll(" ", "").toLowerCase();

// Create a HashMap to store character frequencies

Map<Character, Integer> charFrequency = new HashMap<>();

for (char c : input.toCharArray()) {

if (charFrequency.containsKey(c)) {

charFrequency.put(c, charFrequency.get(c) + 1);

} else {

charFrequency.put(c, 1);

}

}

System.out.println("Character frequencies in the string:");

for (Map.Entry<Character, Integer> entry : charFrequency.entrySet()) {

System.out.println("'" + entry.getKey() + "' - " + entry.getValue() + " times");

}

}

}

Character frequencies in the string:

'!' - 1 times

'r' - 1 times

'd' - 1 times

'e' - 1 times

'w' - 1 times

'h' - 1 times

'l' - 3 times

',' - 1 times

'o' - 2 times

20) Java Program to find the largest and smallest word in a string

public class LargestAndSmallestWords {

public static void main(String[] args) {

String input = "This is a sample sentence with several words of different lengths.";

// Split the input string into words using space as the delimiter

String[] words = input.split(" ");

if (words.length == 0) {

System.out.println("No words found in the input string.");

return;

}

String smallestWord = words[0];

String largestWord = words[0];

for (String word : words) {

// Remove punctuation and convert to lowercase to standardize words

word = word.replaceAll("[^a-zA-Z]", "").toLowerCase();

if (word.length() < smallestWord.length()) {

smallestWord = word;

}

if (word.length() > largestWord.length()) {

largestWord = word;

}

}

System.out.println("Smallest word: " + smallestWord);

System.out.println("Largest word: " + largestWord);

}

}

LargestAndSmallestWords

Smallest word: a

Largest word: different

21) Java Program to find the most repeated word in a text file

import java.io.BufferedReader;

import java.io.FileReader;

import java.util.HashMap;

import java.util.Map;

public class MostRepeatedWordInFile {

public static void main(String[] args) throws Exception {

String fileName = "your\_text\_file.txt"; // Replace with your file path

BufferedReader reader = new BufferedReader(new FileReader(fileName));

String line;

Map<String, Integer> wordFrequency = new HashMap<>();

while ((line = reader.readLine()) != null) {

String[] words = line.split("\\s+");

for (String word : words) {

word = word.toLowerCase().replaceAll("[^a-zA-Z]", "");

if (word.length() > 0) {

wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);

}

}

}

reader.close();

String mostRepeatedWord = null;

int maxFrequency = 0;

for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {

if (entry.getValue() > maxFrequency) {

mostRepeatedWord = entry.getKey();

maxFrequency = entry.getValue();

}

}

System.out.println("Most repeated word: " + mostRepeatedWord);

System.out.println("Frequency: " + maxFrequency);

}

}

22) Java Program to find the number of the words in the given text file

import java.io.BufferedReader;

import java.io.FileReader;

public class WordCountInFile {

public static void main(String[] args) throws Exception {

String fileName = "your\_text\_file.txt"; // Replace with your file path

BufferedReader reader = new BufferedReader(new FileReader(fileName));

String line;

int wordCount = 0;

while ((line = reader.readLine()) != null) {

String[] words = line.split("\\s+");

wordCount += words.length;

}

reader.close();

System.out.println("Number of words in the file: " + wordCount);

}

}

23) Java Program to separate the Individual Characters from a String

public class SeparateCharactersFromString {

public static void main(String[] args) {

String input = "Hello, World!";

for (int i = 0; i < input.length(); i++) {

char character = input.charAt(i);

System.out.println(character);

}

}

}

1. Java Program to swap two string variables without using a third or temp variables.

public class SwapStringsWithoutTempVariable {

public static void main(String[] args) {

String str1 = "Hello";

String str2 = "World";

System.out.println("Before swapping:");

System.out.println("str1 = " + str1);

System.out.println("str2 = " + str2);

str1 = str1 + str2;

str2 = str1.substring(0, str1.length() - str2.length());

str1 = str1.substring(str2.length());

System.out.println("After swapping:");

System.out.println("str1 = " + str1);

System.out.println("str2 = " + str2);

}

}

1. Java Program to print the smallest and biggest possible palindrome word in a given string

public class PalindromeWordsInString {

public static void main(String[] args) {

String input = "racecar is a civic deed, but radar is not a palindrome";

String[] words = input.split("\\s+");

String smallestPalindrome = null;

String largestPalindrome = null;

for (String word : words) {

word = word.toLowerCase().replaceAll("[^a-zA-Z]", "");

if (isPalindrome(word)) {

if (smallestPalindrome == null || word.length() < smallestPalindrome.length()) {

smallestPalindrome = word;

}

if (largestPalindrome == null || word.length() > largestPalindrome.length()) {

largestPalindrome = word;

}

}

}

System.out.println("Smallest palindrome word: " + smallestPalindrome);

System.out.println("Largest palindrome word: " + largestPalindrome);

}

public static boolean isPalindrome(String word) {

int left = 0;

int right = word.length() - 1;

while (left < right) {

if (word.charAt(left) != word.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

}

Java Searching and Sorting Programs

1. Linear Search in Java

class LinearSearch{

public static void main(String[] args) {

int []arr = {25,1,6,3,5,4,48,6,5,2,14};

int key = 14;// found in given array

if(Lsearch(arr, key)){

System.out.println("found in given array ");

}else System.out.println(" not found in given array ");

}

public static Boolean Lsearch(int []arr ,int key) {

for (int i = 0; i < arr.length; i++) {

if (key == arr[i]) {

return true;

}

}

return false;

}

} found in given array

1. Binary Search in Java

public class binary\_Search {

public static void main(String args[]) {

int[] nums = {2, 4, 6, 10, 12, 35, 46, 48, 59, 66, 98};

int key = 66;

int value =BinarySearch(0, nums.length-1,nums,key );

if(value>=0){

System.out.println("We found the value at index "+value);

}else {

System.out.println("Sorry value is not present in array");

}

}

static int BinarySearch(int f,int l,int[]arr, int key) {

//checking for f=0 and last as it is increasing or decreasing according to the key if f is

// greater then l the it return -1.

if (f > l){

return -1;

}

// finding the mid value

int mid = (f+l)/ 2;

// comparing mid is eqaual to key or not if equal then return mid not and run others

if (arr[mid] == key) {

return mid;

}

// if key is smaller mid -1 and assign to last value of the arr as arr end here

else if (key < arr[mid]) {

return BinarySearch(f, mid - 1, arr, key);

}

// if key is greater then mid+1 assign as the index start form here

else {

return BinarySearch(mid + 1, l, arr, key);

}

}}

We found the value at index 9

1. Bubble Sort in Java

class soriting {

public static void main(String args[]){

int [] nums={3,1,2,5,7,6,4};

bubbleSrot(nums);

System.out.println(Arrays.toString(nums));

}

static void bubbleSrot(int[] arr){

for (int i = 0; i < arr.length; i++) {

for (int j = 1; j < arr.length - i; j++) {

if (arr[j] < arr[j - 1]) {

int temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

}

}

}

}

}

[1, 2, 3, 4, 5, 6, 7]

1. Selection Sort in Java

class soriting {

public static void main(String args[]){

int [] nums={3,1,2,5,7,6,4};

SelectionSort(nums);

// bubbleSrot(nums);

// insetionSort(nums);

System.out.println(Arrays.toString(nums));

}

public static void SelectionSort(int []arr){

for (int i = 0; i < arr.length-1; i++) {

int index = i;

for (int j = i+1; j < arr.length; j++) {

if (arr[j] < arr[index])

index = j;

}

swap(arr, i, index);

}

}}

[1, 2, 3, 4, 5, 6, 7]

5) Insertion Sort in Java

class soriting {

public static void main(String args[]){

int [] nums={3,1,2,5,7,6,4};

// bubbleSrot(nums);

insetionSort(nums);

System.out.println(Arrays.toString(nums));

}

static void insetionSort(int []arr){

for (int i = 0; i < arr.length - 1; i++) {

for (int j = i+1; j >0; j--) {

if(arr[j]<arr[j-1]){

swap(arr,j,j-1);

}

else {

break;

}

}

}

}

static void swap(int[]arr,int first,int last){

int temp=arr[first];

arr[first]=arr[last];

arr[last]=temp;

}

}

[1, 2, 3, 4, 5, 6, 7]

Java Conversion Programs

import java.sql.Timestamp;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class assignmentJava {

}

/\* conversions

1) How to convert String to int in Java\*/

class StringToInt{

public static void main(String[] args) {

String s1 = "12345";

int s = Integer.valueOf(s1);

System.out.println(s);

}

}//12345

/\* 2) How to convert int to String in Java \*/

class IntToString{

public static void main(String[] args) {

int value = 1234;

String s1 = String.valueOf(value);

System.out.println(s1);

}

}//1234

/\*3) How to convert String to long in Java \*/

class StringToLong{

public static void main(String[] args) {

String value = "1234";

long s1 = Long.parseLong(value);

System.out.println(s1);

}

}//1234

/\*4) How to convert long to String in Java \*/

class LongToString{

public static void main(String[] args) {

long value = 1234l;

String s1 = String.valueOf(value);

System.out.println(s1);

}

}//1234.0

/\*5) How to convert String to float in Java\*/

class StringToFloat{

public static void main(String[] args) {

String value = "1234";

float f1 = Float.parseFloat(value);

System.out.println(f1);

}

}//1234.0

/\*6) How to convert float to String in Java\*/

class FloatToString{

public static void main(String[] args) {

float value = 1234f;

String s1 = String.valueOf(value);

System.out.println(s1);

}

}//1234.0

/\*7) How to convert String to double in Java\*/

class StringToDouble{

public static void main(String[] args) {

String S1 = "1234";

Double D = Double.parseDouble(S1 );

System.out.println(D);

}

}//1234.0

/\*8) How to convert double to String in Java\*/

class DoubleToString{

public static void main(String[] args) {

Double D = 125.0;

String S1 = String.valueOf(D);

System.out.println(S1);

}

}//125.0

/\*9) How to convert String to Date in Java\*/

class StringToDate {

public static void main(String[] args) throws ParseException {

String name = "12-05-2003";

SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyy");

Date date = dateFormat.parse(name);

System.out.println(date);

}

}//Parsed Date: Mon May 12 00:00:00 IST 2003

// /\*10) How to convert Date to String in Java\*/

class DateToString {

public static void main(String[] args) {

Date date = new Date();

String dateFormate = "dd-MM-YYYY";

SimpleDateFormat dateformat = new SimpleDateFormat(dateFormate);

String Str = dateformat.format(date);

System.out.println(Str);

}

}//02-11-2023

/\*11) How to convert String to char in Java\*/

class StringToCharEx {

public static void main(String[] args) {

String str = "Himanshu";

char c = str.charAt(1);

System.out.println(c);

}

}// i

/\*12) How to convert char to String in Java\*/

class CharToString{

public static void main(String[] args) {

char c = 'Y';

String str = String.valueOf(c);

System.out.println(str);

}

}// Y

/\*13) How to convert String to Object in Java\*/

class StringToObject{

public static void main(String[] args) {

String str = "Himanshu";

Object obj = str;

System.out.println(obj);

}

}// Himanshu

/\*14) How to convert Object to String in Java\*/

class Himanshu{

}

class ObjectToString{

public static void main(String[] args) {

Himanshu him = new Himanshu();

String x = him.toString();

System.out.println(x);

}

}//Himanshu@79698539

/\*15) How to convert int to long in Java\*/

class IntToLong {

public static void main(String[] args) {

int x = 5365;

long y = x;

System.out.println(y);

}

}//5365

/\*16) How to convert long to int in Java\*/

class LongToInt {

public static void main(String[] args) {

long l = 152l;

int i = (int) l;

System.out.println("int value " + i);

}

}//int value 152

/\*17) How to convert int to double in Java\*/

class IntToDouble {

public static void main(String[] args) {

int i = 654;

double doubleValue = i;

System.out.println("double value is " + doubleValue);

}

}//double value is 654.0

/\*18) How to convert double to int in Java\*/

class DoubleToInt {

public static void main(String[] args) {

double doubleValue = 9548890.000;

int intValue = (int) doubleValue;

System.out.println("integer value is " + intValue);

}

}//integer value is 9548890

/\*19) How to convert char to int in Java\*/

class CharToInt {

public static void main(String[] args) {

char ch = 'A';

int x = ch;

System.out.println(x);

}

} //65

/\*20) How to convert int to char in Java\*/

class IntToChar{

public static void main(String[] args) {

int x = 97;

char ch = (char) x;

System.out.println(ch);

}

}// a

/\* 21) How to convert String to boolean in Java\*/

class StringToBoolean {

public static void main(String[] args) {

String str = "True";

boolean b = Boolean.parseBoolean(str);

System.out.println(b);

}

} // true

/\* 22) How to convert boolean to String in Java\*/

class BooleanToString{

public static void main(String[] args) {

boolean x = false;

String y = String.valueOf(x);

System.out.println(y);

}

}// false

/\* 23) How to convert date to Timestamp in Java\*/

class DateToTimeStamp {

public static void main(String[] args) {

Date date = new Date();

Timestamp temp = new Timestamp(date.getTime());

System.out.println(temp);

}

}//2023-11-02 15:07:05.638

/\*24) How to convert timestamp to Date in Java\*/

class TimestampToDate{

public static void main(String[] args) {

long l= System.currentTimeMillis();

Timestamp currenttime = new Timestamp(l);

Date date = new Date(currenttime.getTime());

System.out.println(date);

}

}//Thu Nov 02 15:10:09 IST 2023

/\*25) How to convert binary to Decimal in Java\*/

class BinarytoDecimal{

public static void main(String[] args) {

String str = "10011";

int dec = Integer.parseInt(str,2);

System.out.println("the Decimal number is "+dec);

}

} // the decimal number is 19

/\*26) How to convert decimal to Binary in Java\*/

class DecimalTobinary{

public static void main(String[] args) {

int x = 65;

String y = Integer.toBinaryString(x);

System.out.println("the binary number is "+y);

}

}//the binary number is 1000001

/\*27) How to convert Hex to Decimal in Java\*/

class HexToDecimal{

public static void main(String[] args) {

String hex = "A";

int Binary= Integer.parseInt(hex,16);

System.out.println("the decimal number is "+Binary);

}

}//the decimal number is 10

/\*28) How to convert decimal to Hex in Java\*/

class DecimalToHex{

public static void main(String[] args) {

int x = 65;

String y = Integer.toHexString(x);

System.out.println("the hex number is "+y);

}

} // the hex number is 41

/\*29) How to convert octal to Decimal in Java\*/

class OctalToDecimal{

public static void main(String[] args) {

String octal = "011";

int decimal = Integer.parseInt(octal,8);

System.out.println("the decimal number is "+decimal);

}

}// the decimal number is 9

/\*30) How to convert decimal to Octal in Java\*/

class DecimalToOctal{

public static void main(String[] args) {

int decimal = 54;

String octal = Integer.toOctalString(decimal);

System.out.println("the octal value is "+octal);

}

} // the octal value is 66

Java Singly Linked List Programs

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

public SinglyLinkedList() {

head = null;

}

public void add(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public int countNodes() {

int count = 0;

Node current = head;

while (current != null) {

count++;

current = current.next;

}

return count;

}

public void display() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public void displayReverse(){

reverse(head);

}

private void reverse(Node node){

if(node == null){

return;

}

displayReverse();

System.out.println(node.data+" ");

}

}

/\*1) Singly linked list Examples in Java \*/

class LinkedList1 {

public static void main(String[] args) {

SinglyLinkedList llist = new SinglyLinkedList();

llist.add(12);

llist.add(25);

llist.add(85);

llist.add(66);

llist.display();

}

}

//12 -> 25 -> 85 -> 66 -> null

/\* 2) Java Program to create and display a singly linked list\*/

class SinglyLinkedList1 {

public static void main(String[] args) {

LinkedList<Integer> SinglyLinkedList = new LinkedList<>();

SinglyLinkedList.add(25);

SinglyLinkedList.add(68);

System.out.println(SinglyLinkedList);

}

}//[25, 68]

/\* 3) Java program to create a singly linked list of n nodes and count the number of nodes\*/

class him{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

Scanner scanner = new Scanner(System.in);

System.out.print("enter the value of (n) node : ");

int n = scanner.nextInt();

for (int i = 0; i < n; i++) {

System.out.print("enter the "+i+" value data :");

int data = scanner.nextInt();

list.add(i,data);

}

System.out.println("total number of node is "+list.size());

}

}/\* enter the value of (n) node : 4

enter the 0 value data :2

enter the 1 value data :3

enter the 2 value data :4

enter the 3 value data :6

total number of node is 4

\*/

/\* 4) Java program to create a singly linked list of n nodes and display it in reverse order\*/

class MReverseList {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

LinkedList<Integer> list = new LinkedList<>();

System.out.print("enter the value of (n) node : ");

int n = scanner.nextInt();

for (int i = 0; i < n; i++) {

System.out.print("enter the "+i+" value data :");

int data = scanner.nextInt();

list.add(i,data);

}

System.out.print("list in reverse order : ");

for (int i = list.size()-1; i >= 0; i--){

System.out.print(" "+list.get(i));

}

}

}/\*enter the 0 value data :1

enter the 1 value data :2

enter the 2 value data :3

enter the 3 value data :4

list in reverse order : 4 3 2 1\*/

/\* 5) Java program to delete a node from the beginning of the singly linked list\*/

class DeleteFromBegin{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

System.out.println("linked list value before delete "+list);

list.remove();

System.out.println("linked list value After delete "+list);

}

}/\*linked list value before delete [12, 24, 33, 44]

linked list value After delete [24, 33, 44]\*/

/\* 6) Java program to delete a node from the middle of the singly linked list\*/

class DeleteFromMiddle {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before delete " + list);

list.remove(list.size()/2);

System.out.println("linked list value After delete middle element " + list);

}

}/\*linked list value before delete [12, 24, 33, 44, 63]

linked list value After delete middle element [12, 24, 44, 63]\*/

/\* 7) Java program to delete a node from the end of the singly linked list\*/

class DeleteFromLast {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

// int x = list.size();

System.out.println("linked list value before delete " + list);

list.remove(list.size()-1);

System.out.println("linked list value After delete last element " + list);

}

}/\*linked list value before delete [12, 24, 33, 44, 63]

linked list value After delete last element [12, 24, 33, 44]\*/

/\* 8) Java program to determine whether a singly linked list is the palindrome\*/

class PalindromeOrNot{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(1);

list.add(2);

list.add(2);

list.add(1);

if (YesOrNot(list)){

System.out.println("LinkedList is Palindrome ");

}

else System.out.println("LinkedList is not palindrome");

}

public static boolean YesOrNot(LinkedList <Integer>list){

int left = 0 ;

int right = list.size()-1;

while (left<right){

if (!list.get(left).equals(list.get(right))){

return false;

}

left++;

right--;

}

return true;

}

}//LinkedList is Palindrome

/\* 9) Java program to find the maximum and minimum value node from a linked list\*/

class MaxMinNode {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(52);

list.add(33);

list.add(44);

list.add(43);

System.out.println("element in linked list" + list);

int big =0;

int node =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)>big){

big = list.get(i);

node=i;

}

}

System.out.println("Node of the biggest element is "+node);

int small =list.get(0);

int node1 =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)<small){

small = list.get(i);

node1=i;

}

}

System.out.println("Node of the smallest element is "+node1);

}

}/\*element in linked list[12, 52, 33, 44, 43]

Node of the biggest element is 1

Node of the smallest element is 0\*/

/\* 10) Java Program to insert a new node at the middle of the singly linked list\*/

class InsertNewNodeAtMiddle {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at middle " + list);

list.add(list.size()/2,22);

System.out.println("linked list value After adding element at middle " + list);

list.add(list.size()/2,222);

System.out.println("linked list value After adding element at middle " + list);

}

}/\*linked list value before adding at middle [12, 24, 33, 44, 63]

linked list value After adding element at middle [12, 24, 22, 33, 44, 63]

linked list value After adding element at middle [12, 24, 22, 222, 33, 44, 63]\*/

/\* 11) Java program to insert a new node at the beginning of the singly linked list\*/

class InsertNewNodeAtBeginning {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at beginning " + list);

list.add(0,22);

System.out.println("linked list value After adding element at beginning " + list);

list.add(0,222);

System.out.println("linked list value After adding element at beginning " + list);

}

}/\*linked list value before adding at beginning [12, 24, 33, 44, 63]

linked list value After adding element at beginning [22, 12, 24, 33, 44, 63]

linked list value After adding element at beginning [222, 22, 12, 24, 33, 44, 63]\*/

/\* 12) Java program to insert a new node at the end of the singly linked list\*/

class InsertNewNodeAtEnd {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at beginning " + list);

list.add(list.size(),22);

System.out.println("linked list value After adding element at beginning " + list);

list.add(list.size(),222);

System.out.println("linked list value After adding element at beginning " + list);

}

}/\*linked list value before adding at beginning [12, 24, 33, 44, 63]

linked list value After adding element at beginning [ 12, 24, 33, 44, 63,22]

linked list value After adding element at beginning [ 12, 24, 33, 44, 63, 22, 222]\*/

/\* 13) Java program to remove duplicate elements from a singly linked list\*/

class RemoveDuplicate{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

list.add(25);

list.add(25);

list.add(32);

list.add(21);

list.add(36);

System.out.println("list with duplicate value"+list);

Set<Integer> set =new HashSet<>(list);

ArrayList<Integer> WithoutDuplication =new ArrayList<>(set);

System.out.println("List without any Duplicate value "+ WithoutDuplication);

}

}//list with duplicate value[25, 32, 21, 36, 25, 25, 32, 21, 36]

// List without any Duplicate value [32, 36, 21, 25]

/\* 14) Java Program to search an element in a singly linked list\*/

class SearchElement{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

int key = 36;

if (Search(list,key)){

System.out.println("Element found in the list");

}

else {

System.out.println("Element not found in the list");

}

}//Element found in the list

public static boolean Search(LinkedList<Integer> list, int key){

for (int i = 0; i < list.size(); i++) {

if (list.get(i)==key) {

return true;

}

}

return false;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Java Circular Linked List Programs\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) Java program to create and display a Circular Linked List\*/

class cirularLinkedList {

Node2 head;

Node2 tail;

public cirularLinkedList(){

this.head= null;

this.tail= null;

}

public void Insert(int val){

Node2 node = new Node2(val);

if(head == null){

head = node;

tail = node;

return;

}

tail.next = node;

node.next = head;

tail = node;

}

public void InsertValue(int val){

Node2 node = new Node2(val);

if(head == null){

head = node;

tail = node;

return;

}

tail.next = node;

node.next = head;

tail = node;

}

public void display(){

Node2 node = head;

if (head != null){

do {

System.out.print(node.val+" --> ");

node = node.next;

}while (node != head);

}

System.out.println("end");

}

public void display2() {

Node2 current = head;

do {

System.out.print(current.val + " ");

current = current.next;

} while (current != head);

System.out.println();

}

public void bubbleSort2() {

if (head == null) {

System.out.println("List is empty.");

return;

}

Node2 current;

Node2 tail = null;

boolean swapped;

do {

swapped = false;

current = head;

while (current.next != tail) {

if (current.val > current.next.val) {

int temp = current.val;

current.val = current.next.val;

current.next.val = temp;

swapped = true;

}

current = current.next;

}

tail = current;

} while (swapped);

}

}

class Node2 {

int val;

Node2 next;

public Node2(int val) {

this.val = val;

// this.next = next;

}

}

class CllDisp{

public static void main(String[] args) {

cirularLinkedList list = new cirularLinkedList();

list.Insert(25);

list.Insert(62);

list.Insert(52);

list.Insert(99);

list.Insert(58);

list.display();

}

}

//25 --> 62 --> 52 --> 99 --> 58 --> end

/\*

2) Java program to create a Circular Linked List of N nodes and count the number of nodes\*/

class CountNodeCll{

public static void main(String[] args) {

cirularLinkedList list = new cirularLinkedList();

Scanner sc = new Scanner(System.in);

System.out.print("enter the value of node (n) :- ");

int nodeValue = sc.nextInt();

int count=0;

for (int i = 0; i < nodeValue; i++) {

System.out.print("enter the value of "+i+" index :- ");

int UserInput = sc.nextInt();

list.Insert(UserInput);

count++;

}

list.display();

System.out.println("total number of nodes are : - "+count);

}

}

/\*

3) Java program to create a Circular Linked List of n nodes and display it in reverse order \*/

class ReverseCll{

public static void main(String[] args) {

LinkedList<String> list = new LinkedList<>();

list.offer("First");

list.add("Second");

list.add("Third");

list.offer("Four");

ListIterator<String> Reverse = list.listIterator(list.size());

while (Reverse.hasPrevious()){

System.out.print(Reverse.previous()+" --> ");

}

System.out.println("end");

}

}

//Four --> Third --> Second --> First --> end

/\*4) Java program to delete a node from the beginning of the Circular Linked List\*/

class DeleteNodeOfFirst{

public static void main(String[] args) {

LinkedList<Integer> CLL =new LinkedList<>();

CLL.offer(10);

CLL.offer(20);

CLL.offer(30);

CLL.offer(40);

System.out.println(CLL);

System.out.println("Remove "+CLL.pollFirst());

System.out.println(CLL);

}

}

//[10, 20, 30, 40]

//Remove 10

//[20, 30, 40]

/\*5) Java program to delete a node from the end of the Circular Linked List \*/

class DeleteNodeOfEnd{

public static void main(String[] args) {

LinkedList<Integer> CLL =new LinkedList<>();

CLL.offer(10);

CLL.offer(20);

CLL.offer(30);

CLL.offer(40);

CLL.offer(50);

System.out.println(CLL);

System.out.println("Remove "+CLL.pollLast());

System.out.println(CLL);

}

}

//[10, 20, 30, 40, 50]

//Remove 50

//[10, 20, 30, 40]

/\*6) Java program to delete a node from the middle of the Circular Linked List\*/

class DeleteMiddleNode{

public static void main(String[] args) {

LinkedList<Integer> CLL =new LinkedList<>();

CLL.offer(10);

CLL.offer(20);

CLL.offer(30);

CLL.offer(40);

CLL.offer(50);

System.out.println(CLL);

System.out.println("Remove middle "+CLL.remove(CLL.size()/2));

System.out.println(CLL);

}

}

//

//[10, 20, 30, 40, 50]

// Remove middle 30

// [10, 20, 40, 50]

/\*7) Java program to find the maximum and minimum value node from a circular linked list\*/

class MaxMinNodeCLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(52);

list.add(33);

list.add(64);

list.add(43);

System.out.println("element in linked list" + list);

int big =0;

int node =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)>big){

big = list.get(i);

node=i;

}

}

System.out.println("Node of the biggest element is "+node);

int small =list.get(0);

int node1 =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)<small){

small = list.get(i);

node1=i;

}

}

System.out.println("Node of the smallest element is "+node1);

}

}

//element in linked list[12, 52, 33, 64, 43]

//Node of the biggest element is 3

//Node of the smallest element is 0

/\*8) Java program to insert a new node at the beginning of the Circular Linked List\*/

class AddElementAtEndOfCll{

public static void main(String[] args) {

cirularLinkedList list = new cirularLinkedList();

Scanner sc = new Scanner(System.in);

System.out.print("enter the value of node (n) :- ");

int nodeValue = sc.nextInt();

int count=0;

for (int i = 0; i < nodeValue; i++) {

System.out.print("enter the value of "+i+" index :- ");

int UserInput = sc.nextInt();

list.Insert(UserInput);

count++;

}

list.display();

list.Insert(35);

System.out.println("Adding element at last of the linked list : ");

list.display();

}

}

//enter the value of node (n) :- 6

//enter the value of 0 index :- 53

//enter the value of 1 index :- 22

//enter the value of 2 index :- 33

//enter the value of 3 index :- 14

//enter the value of 4 index :- 32

//enter the value of 5 index :- 55

//53 --> 22 --> 33 --> 14 --> 32 --> 55 --> end

//Adding element at last of the linked list :

//53 --> 22 --> 33 --> 14 --> 32 --> 55 --> 35 --> end

/\*9) Java program to insert a new node at the end of the Circular Linked List \*/

class InsertNodeAtLastCLL{

public static void main(String[] args) {

LinkedList<Integer> CLL =new LinkedList<>();

CLL.offer(10);

CLL.offer(20);

CLL.offer(30);

CLL.offer(40);

CLL.offer(50);

System.out.println(CLL);

System.out.println("Insert element At first node "+CLL.offerFirst(69));

System.out.println(CLL);

}

}

//[10, 20, 30, 40, 50]

//Insert element At first node true

//[69, 10, 20, 30, 40, 50]

/\*

10) Java program to insert a new node at the middle of the Circular Linked List\*/

class DeleteMiddleNodeCLL1{

public static void main(String[] args) {

LinkedList<Integer> CLL =new LinkedList<>();

CLL.offer(10);

CLL.offer(20);

CLL.offer(30);

CLL.offer(40);

CLL.offer(50);

System.out.println(CLL);

CLL.add(CLL.size()/2,69);

System.out.println("After adding middle element : ");

System.out.println(CLL);

}

}

//[10, 20, 30, 40, 50]

//After adding middle element :

//[10, 20, 69, 30, 40, 50]

/\*

11) Java program to remove duplicate elements from a Circular Linked List

\*/

class RemoveDuplicateCLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

list.add(25);

list.add(25);

list.add(32);

list.add(21);

list.add(36);

System.out.println("list with duplicate value"+list);

Set<Integer> set =new HashSet<>(list);

ArrayList<Integer> WithoutDuplication =new ArrayList<>(set);

System.out.println("List without any Duplicate value "+ WithoutDuplication);

}

}

//list with duplicate value[25, 32, 21, 36, 25, 25, 32, 21, 36]

// List without any Duplicate value [32, 36, 21, 25]

/\*12) Java program to search an element in a Circular Linked List\*/

class SearchElementCLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

int key = 36;

if (Search(list,key)){

System.out.println("Element found in the list");

}

else {

System.out.println("Element not found in the list");

}

}//Element found in the list

public static boolean Search(LinkedList<Integer> list, int key){

for (int i = 0; i < list.size(); i++) {

if (list.get(i)==key) {

return true;

}

}

return false;

}

}

/\*13) Java program to sort the elements of the Circular Linked List\*/

class SortCLL{

public static void main(String[] args) {

cirularLinkedList list = new cirularLinkedList();

list.Insert(3);

list.Insert(2);

list.Insert(1);

list.display();

list.bubbleSort2();

list.display2();

list.display();

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Java Doubly Linked List Programs\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) Java program to convert a given binary tree to a doubly linked list\*/

/\*

2) Java program to create a doubly linked list from a ternary tree

/\*3) Java program to create a doubly linked list of n nodes and count the number of nodes\*/

class Nnodes{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

Scanner scanner = new Scanner(System.in);

System.out.print("enter the value of (n) node : ");

int n = scanner.nextInt();

for (int i = 0; i < n; i++) {

System.out.print("enter the "+i+" value data :");

int data = scanner.nextInt();

list.add(i,data);

}

System.out.println("total number of node is "+list.size());

}

}/\*

enter the value of (n) node : 6

enter the 0 value data :2

enter the 1 value data :3

enter the 2 value data :6

enter the 3 value data :2

enter the 4 value data :4

enter the 5 value data :5

total number of node is 6

\*/

/\*4) Java program to create a doubly linked list of n nodes and display it in reverse order\*/

class DReverseList {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

LinkedList<Integer> list = new LinkedList<>();

System.out.print("enter the value of (n) node : ");

int n = scanner.nextInt();

for (int i = 0; i < n; i++) {

System.out.print("enter the "+i+" value data :");

int data = scanner.nextInt();

list.add(i,data);

}

System.out.print("list in reverse order : ");

for (int i = list.size()-1; i >= 0; i--){

System.out.print(" "+list.get(i));

}

}

}/\*enter the value of (n) node : 6

enter the 0 value data :33

enter the 1 value data :2

enter the 2 value data :1

enter the 3 value data :3

enter the 4 value data :4

enter the 5 value data :7

list in reverse order : 7 4 3 1 2 33

\*/

/\*

5) Java program to create and display a doubly linked list\*/

class DLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(2);

list.add(65);

list.add(58);

System.out.println("Doubly linkedList has value in it :-"+list);

list.offer(65);

System.out.println(list);

}

}/\*Doubly linkedList has value in it :-[2, 65, 58]

[2, 65, 58, 65]\*/

/\*6) Java program to delete a new node from the beginning of the doubly linked list\*/

class DeleteFromBeginOfDLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

System.out.println("doubly linked list value before delete first element "+list);

list.pollFirst();//using pool first method/function that delete the first element form the list

System.out.println("doubly linked list value After delete first element " +list);

}

}//doubly linked list value before delete first element [12, 24, 33, 44]

// doubly linked list value After delete first element [24, 33, 44]

/\*7) Java program to delete a new node from the end of the doubly linked list\*/

class DeleteFromLastDLL {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before delete " + list);

list.pollLast();

System.out.println("linked list value After deleted last element " + list);

}

}//linked list value before delete [12, 24, 33, 44, 63]

// linked list value After deleted last element [12, 24, 33, 44]

/\*8) Java program to delete a new node from the middle of the doubly linked list \*/

class DeleteFromMiddleDll {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before delete " + list);

list.remove(list.size() / 2);

System.out.println("linked list value After delete middle element " + list);

}

}

//linked list value before delete [12, 24, 33, 44, 63]

//linked list value After delete middle element [12, 24, 44, 63]

/\*9) Java program to find the maximum and minimum value node from a doubly linked list\*/

class MaxMinNodeDLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(52);

list.add(33);

list.add(44);

list.add(43);

System.out.println("element in linked list" + list);

int big =0;

int node =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)>big){

big = list.get(i);

node=i;

}

}

System.out.println("Node of the biggest element is "+node);

int small =list.get(0);

int node1 =0;

for (int i = 0; i < list.size(); i++) {

if (list.get(i)<small){

small = list.get(i);

node1=i;

}

}

System.out.println("Node of the smallest element is "+node1);

}

}

//element in linked list[12, 52, 33, 44, 43]

//Node of the biggest element is 1

//Node of the smallest element is 0

/\*10) Java program to insert a new node at the beginning of the Doubly Linked list\*/

class InsertNewNodeAtBeginningDll {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at beginning " + list);

list.offerFirst(986);

System.out.println("linked list value After adding element at beginning " + list);

list.offerFirst(253);

System.out.println("linked list value After adding element at beginning " + list);

}

}

//linked list value before adding at beginning [12, 24, 33, 44, 63]

//linked list value After adding element at beginning [986, 12, 24, 33, 44, 63]

//linked list value After adding element at beginning [253, 986, 12, 24, 33, 44, 63]

/\*10) Java program to insert a new node at the end of the Doubly Linked List\*/

class InsertNewNodeAtEndDll {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at last " + list);

list.offerLast(253);

System.out.println("linked list value After adding element at last " + list);

list.offerLast(256);

System.out.println("linked list value After adding element at last " + list);

}

}

//linked list value before adding at last [12, 24, 33, 44, 63]

//linked list value After adding element at last [12, 24, 33, 44, 63, 253]

//linked list value After adding element at last [12, 24, 33, 44, 63, 253, 256]

/\*12) Java program to insert a new node at the middle of the Doubly Linked List\*/

class InsertNewNodeAtMiddleDll {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(12);

list.add(24);

list.add(33);

list.add(44);

list.add(63);

System.out.println("linked list value before adding at middle " + list);

list.add(list.size() / 2, 22);

System.out.println("linked list value After adding element at middle " + list);

list.add(list.size() / 2, 222);

System.out.println("linked list value After adding element at middle " + list);

}

}

//linked list value before adding at middle [12, 24, 33, 44, 63]

//linked list value After adding element at middle [12, 24, 22, 33, 44, 63]

//linked list value After adding element at middle [12, 24, 22, 222, 33, 44, 63]

/\*13) Java program to remove duplicate elements from a Doubly Linked List\*/

class RemoveDuplicateDll {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

list.add(25);

list.add(25);

list.add(32);

list.add(21);

list.add(36);

System.out.println("list with duplicate value" + list);

Set<Integer> set = new HashSet<>(list);

ArrayList<Integer> WithoutDuplication = new ArrayList<>(set);

System.out.println("List without any Duplicate value " + WithoutDuplication);

}

}

//list with duplicate value[25, 32, 21, 36, 25, 25, 32, 21, 36]

// List without any Duplicate value [32, 36, 21, 25]

/\*14) Java program to rotate doubly linked list by N nodes\*/

class Node1 {

int data;

Node1 next;

Node1 prev;

Node1(int data) {

this.data = data;

next = null;

prev = null;

}

}

class DLLLinkedList {

Node1 head;

DLLLinkedList() {

head = null;

}

// Function to add a node at the end of the linked list

void addNode(int data) {

Node1 newNode = new Node1(data);

if (head == null) {

head = newNode;

} else {

Node1 temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

newNode.prev = temp;

}

}

// Function to rotate the linked list by N nodes

void rotateByN(int n) {

if (head == null || n <= 0) {

return;

}

Node1 current = head;

int count = 1;

while (count < n && current != null) {

current = current.next;

count++;

}

if (current == null) {

return; // The linked list is shorter than N

}

Node1 NthNode = current;

while (current.next != null) {

current = current.next;

}

current.next = head;

head.prev = current;

head = NthNode.next;

head.prev = null;

NthNode.next = null;

}

void bubbleSort() {

if (head == null) {

return;

}

boolean swapped;

Node1 last = null;

do {

swapped = false;

Node1 current = head;

while (current.next != last) {

if (current.data > current.next.data) {

// Swap the data of the current and next nodes

int temp = current.data;

current.data = current.next.data;

current.next.data = temp;

swapped = true;

}

current = current.next;

}

last = current;

} while (swapped);

}

// Function to display the linked list

void display() {

Node1 temp = head;

while (temp != null) {

System.out.print(temp.data + " ");

temp = temp.next;

}

System.out.println();

}

public static void main(String[] args) {

DLLLinkedList list = new DLLLinkedList();

list.addNode(1);

list.addNode(2);

list.addNode(3);

list.addNode(4);

list.addNode(5);

System.out.println("Original Linked List:");

list.display();

int n = 2;

list.rotateByN(n);

System.out.println("Linked List after rotating by " + n + " nodes:");

list.display();

}

}

**Output:-**

//Original Linked List:

//1 2 3 4 5

//Linked List after rotating by 2 nodes:

//3 4 5 1 2

**/\*15) Java program to search an element in a doubly linked list\*/**

class SearchElementInDLL{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<>();

list.add(25);

list.add(32);

list.add(21);

list.add(36);

int key = 36;

if (Search(list,key)){

System.out.println("Element found in the list");

}

else {

System.out.println("Element not found in the list");

}

}//Element found in the list

public static boolean Search(LinkedList<Integer> list, int key){

for (int i = 0; i < list.size(); i++) {

if (list.get(i)==key) {

return true;

}

}

return false;

}

}

/\*

**16) Java program to sort the elements of the doubly linked list**

**\*/**

class SortDll{

public static void main(String[] args) {

DLLLinkedList list = new DLLLinkedList();

list.addNode(5);

list.addNode(3);

list.addNode(8);

list.addNode(1);

list.addNode(9);

System.out.println("Original Doubly Linked List:");

list.display();

list.bubbleSort();

System.out.println("Doubly Linked List after sorting:");

list.display();

}

}

**Output:-**

//Original Doubly Linked List:

//5 3 8 1 9

//Doubly Linked List after sorting:

//1 3 5 8 9

/\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Java Tree Programs\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1) Java Program to calculate the Difference between the Sum of the Odd Level and the Even Level Nodes of a Binary Tree\*/**

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public int calculateLevelDifference() {

return calculateLevelDifference(root, 1);

}

private int calculateLevelDifference(TreeNode node, int level) {

if (node == null) {

return 0;

}

if (level % 2 == 1) { // Odd level

return node.data - calculateLevelDifference(node.left, level + 1) - calculateLevelDifference(node.right, level + 1);

} else { // Even level

return calculateLevelDifference(node.left, level + 1) + calculateLevelDifference(node.right, level + 1) - node.data;

}

}

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.left.right = new TreeNode(5);

tree.root.right.left = new TreeNode(6);

tree.root.right.right = new TreeNode(7);

int levelDifference = tree.calculateLevelDifference();

System.out.println("Difference between the sum of odd level and even level nodes: " + levelDifference);

}

}

**/\*2) Java program to construct a Binary Search Tree and perform deletion and In-order traversal\*/**

class Node {

int data;

Node left;

Node right;

public Node(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinarySearchTree {

private Node root;

public BinarySearchTree() {

root = null;

}

public void insert(int data) {

root = insertRec(root, data);

}

private Node insertRec(Node root, int data) {

if (root == null) {

root = new Node(data);

return root;

}

if (data < root.data) {

root.left = insertRec(root.left, data);

} else if (data > root.data) {

root.right = insertRec(root.right, data);

}

return root;

}

public void delete(int data) {

root = deleteRec(root, data);

}

private Node deleteRec(Node root, int data) {

if (root == null) {

return root;

}

if (data < root.data) {

root.left = deleteRec(root.left, data);

} else if (data > root.data) {

root.right = deleteRec(root.right, data);

} else {

if (root.left == null) {

return root.right;

} else if (root.right == null) {

return root.left;

}

root.data = minValue(root.right);

root.right = deleteRec(root.right, root.data);

}

return root;

}

private int minValue(Node node) {

int minValue = node.data;

while (node.left != null) {

minValue = node.left.data;

node = node.left;

}

return minValue;

}

public void inOrderTraversal() {

inOrderRec(root);

}

private void inOrderRec(Node root) {

if (root != null) {

inOrderRec(root.left);

System.out.print(root.data + " ");

inOrderRec(root.right);

}

}

}

class Main {

public static void main(String[] args) {

BinarySearchTree bst = new BinarySearchTree();

bst.insert(50);

bst.insert(30);

bst.insert(70);

bst.insert(20);

bst.insert(40);

bst.insert(60);

bst.insert(80);

System.out.println("In-order traversal of the BST:");

bst.inOrderTraversal();

System.out.println("\n\nDeleting 30 from the BST:");

bst.delete(30);

bst.inOrderTraversal();

}

}

Output:-

In-order traversal of the BST:

20 30 40 50 60 70 80

Deleting 30 from the BST:

20 40 50 60 70 80

**/\*3) Java program to convert Binary Tree to Binary Search Tree\*/**

import java.util.ArrayList;

import java.util.Collections;

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public void inOrderTraversal(TreeNode node, ArrayList<Integer> list) {

if (node != null) {

inOrderTraversal(node.left, list);

list.add(node.data);

inOrderTraversal(node.right, list);

}

}

public TreeNode convertToBST() {

ArrayList<Integer> nodeList = new ArrayList<>();

inOrderTraversal(root, nodeList);

Collections.sort(nodeList);

return buildBST(nodeList, 0, nodeList.size() - 1);

}

private TreeNode buildBST(ArrayList<Integer> nodeList, int start, int end) {

if (start > end) {

return null;

}

int mid = (start + end) / 2;

TreeNode newNode = new TreeNode(nodeList.get(mid));

newNode.left = buildBST(nodeList, start, mid - 1);

newNode.right = buildBST(nodeList, mid + 1, end);

return newNode;

}

public void inOrderTraversal(TreeNode node) {

if (node != null) {

inOrderTraversal(node.left);

System.out.print(node.data + " ");

inOrderTraversal(node.right);

}

}

}

public class Main {

public static void main(String[] args) {

BinaryTree binaryTree = new BinaryTree();

binaryTree.root = new TreeNode(4);

binaryTree.root.left = new TreeNode(2);

binaryTree.root.right = new TreeNode(6);

binaryTree.root.left.left = new TreeNode(1);

binaryTree.root.left.right = new TreeNode(3);

System.out.println("In-order traversal of the original Binary Tree:");

binaryTree.inOrderTraversal(binaryTree.root);

System.out.println();

TreeNode bstRoot = binaryTree.convertToBST();

System.out.println("In-order traversal of the converted Binary Search Tree:");

binaryTree.inOrderTraversal(bstRoot);

}

}

**Output:-**

In-order traversal of the original Binary Tree:

1 2 3 4 6

In-order traversal of the converted Binary Search Tree:

1 2 3 4 6

4) **Java program to determine whether all leaves are at the same level**

import java.util.LinkedList;

import java.util.Queue;

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public boolean areLeavesAtSameLevel() {

if (root == null) {

return true; // An empty tree has leaves at the same level by definition.

}

Queue<TreeNode> queue = new LinkedList<>();

queue.offer(root);

int leafLevel = -1; // Initialize with an invalid level

while (!queue.isEmpty()) {

int size = queue.size();

for (int i = 0; i < size; i++) {

TreeNode node = queue.poll();

if (node.left == null && node.right == null) {

// Found a leaf node.

if (leafLevel == -1) {

leafLevel = size; // Record the level of the first leaf node.

} else if (size != leafLevel) {

return false; // Leaf node at a different level.

}

}

if (node.left != null) {

queue.offer(node.left);

}

if (node.right != null) {

queue.offer(node.right);

}

}

}

return true; // All leaves are at the same level.

}

}

public class Main {

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

// Construct a binary tree with leaves at the same level.

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.left.right = new TreeNode(5);

tree.root.right.left = new TreeNode(6);

boolean sameLevel = tree.areLeavesAtSameLevel();

System.out.println("Are leaves at the same level? " + sameLevel);

// Construct a binary tree with leaves at different levels.

BinaryTree tree2 = new BinaryTree();

tree2.root = new TreeNode(1);

tree2.root.left = new TreeNode(2);

tree2.root.left.left = new TreeNode(3);

tree2.root.left.left.left = new TreeNode(4);

tree2.root.left.left.right = new TreeNode(5);

sameLevel = tree2.areLeavesAtSameLevel();

System.out.println("Are leaves at the same level? " + sameLevel);

}

}

**Output:-**

Are leaves at the same level? true

Are leaves at the same level? True

1. **Java program to determine whether two trees are identical**

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public boolean areIdenticalTrees(TreeNode root1, TreeNode root2) {

if (root1 == null && root2 == null) {

return true; // Both trees are empty or have reached the leaves.

}

if (root1 != null && root2 != null) {

// Check if the current nodes have the same data and their left and right subtrees are identical.

return (root1.data == root2.data) &&

areIdenticalTrees(root1.left, root2.left) &&

areIdenticalTrees(root1.right, root2.right);

}

return false; // One of the trees is empty while the other is not.

}

}

public class Main {

public static void main(String[] args) {

BinaryTree tree1 = new BinaryTree();

tree1.root = new TreeNode(1);

tree1.root.left = new TreeNode(2);

tree1.root.right = new TreeNode(3);

tree1.root.left.left = new TreeNode(4);

tree1.root.left.right = new TreeNode(5);

BinaryTree tree2 = new BinaryTree();

tree2.root = new TreeNode(1);

tree2.root.left = new TreeNode(2);

tree2.root.right = new TreeNode(3);

tree2.root.left.left = new TreeNode(4);

tree2.root.left.right = new TreeNode(5);

BinaryTree tree3 = new BinaryTree();

tree3.root = new TreeNode(1);

tree3.root.left = new TreeNode(2);

tree3.root.right = new TreeNode(3);

boolean identical1 = tree1.areIdenticalTrees(tree1.root, tree2.root);

System.out.println("Are tree1 and tree2 identical? " + identical1);

boolean identical2 = tree1.areIdenticalTrees(tree1.root, tree3.root);

System.out.println("Are tree1 and tree3 identical? " + identical2);

}

}

**Output:-**

Are tree1 and tree2 identical? true

Are tree1 and tree3 identical? false

1. **Java program to find the maximum width of a binary tree**

import java.util.LinkedList;

import java.util.Queue;

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public int maxWidth() {

if (root == null) {

return 0; // Empty tree has zero width.

}

int maxWidth = 0;

Queue<TreeNode> queue = new LinkedList<>();

queue.offer(root);

while (!queue.isEmpty()) {

int levelWidth = queue.size();

if (levelWidth > maxWidth) {

maxWidth = levelWidth;

}

for (int i = 0; i < levelWidth; i++) {

TreeNode node = queue.poll();

if (node.left != null) {

queue.offer(node.left);

}

if (node.right != null) {

queue.offer(node.right);

}

}

}

return maxWidth;

}

}

public class Main {

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.left.right = new TreeNode(5);

tree.root.right.right = new TreeNode(6);

tree.root.left.left.left = new TreeNode(7);

int maxWidth = tree.maxWidth();

System.out.println("Maximum width of the binary tree: " + maxWidth);

}

}

**Output:-**

Maximum width of the binary tree: 3

1. **Java program to find the largest element in a Binary Tree**

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public int findLargest() {

return findLargest(root);

}

private int findLargest(TreeNode node) {

if (node == null) {

throw new IllegalArgumentException("Tree is empty");

}

while (node.right != null) {

node = node.right;

}

return node.data;

}

}

class Main1 {

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

tree.root = new TreeNode(10);

tree.root.left = new TreeNode(5);

tree.root.right = new TreeNode(15);

tree.root.left.left = new TreeNode(3);

tree.root.left.right = new TreeNode(7);

int largest = tree.findLargest();

System.out.println("Largest element in the binary tree: " + largest);

}

}

**Output:-**

/Largest element in the binary tree: 15

**// 8) Java program to find the maximum depth or height of a tree:**

class TreeNode1 {

int data;

TreeNode left;

TreeNode right;

public TreeNode1(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree1 {

TreeNode root;

public BinaryTree1() {

root = null;

}

public int findMaxDepth() {

return findMaxDepth(root);

}

private int findMaxDepth(TreeNode node) {

if (node == null) {

return 0;

} else {

int leftDepth = findMaxDepth(node.left);

int rightDepth = findMaxDepth(node.right);

return Math.max(leftDepth, rightDepth) + 1;

}

}

}

class Main2 {

public static void main(String[] args) {

BinaryTree1 tree = new BinaryTree1();

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.left.left.left = new TreeNode(5);

int maxDepth = tree.findMaxDepth();

System.out.println("Maximum depth of the tree: " + maxDepth);

}

}

**Output:-**

//Maximum depth of the tree: 4

**// 9) Java program to find the nodes which are at the maximum distance in a Binary Tree:**

class TreeNode2 {

int data;

TreeNode left;

TreeNode right;

public TreeNode2(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree2 {

TreeNode root;

public BinaryTree2() {

root = null;

}

public void findNodesAtMaxDistance() {

findNodesAtMaxDistance(root);

}

private void findNodesAtMaxDistance(TreeNode node) {

if (node == null) {

return;

}

// Do something with the nodes at the maximum distance, e.g., print their data.

System.out.println("Node at maximum distance: " + node.data);

findNodesAtMaxDistance(node.left);

findNodesAtMaxDistance(node.right);

}

}

class Main3 {

public static void main(String[] args) {

BinaryTree2 tree = new BinaryTree2();

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.right.right = new TreeNode(5);

System.out.println("Nodes at the maximum distance in the tree:");

tree.findNodesAtMaxDistance();

}

}

**Output:-**

//Nodes at the maximum distance in the tree:

// Node at maximum distance: 1

// Node at maximum distance: 2

// Node at maximum distance: 4

// Node at maximum distance: 3

// Node at maximum distance: 5

**// 10) Java program to find the smallest element in a tree:**

**Program:-**

class TreeNode3 {

int data;

TreeNode left;

TreeNode right;

public TreeNode3(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree3 {

TreeNode root;

public BinaryTree3() {

root = null;

}

public int findSmallest() {

return findSmallest(root);

}

private int findSmallest(TreeNode node) {

if (node == null) {

throw new IllegalArgumentException("Tree is empty");

}

while (node.left != null) {

node = node.left;

}

return node.data;

}

}

class Main4 {

public static void main(String[] args) {

BinaryTree3 tree = new BinaryTree3();

tree.root = new TreeNode(10);

tree.root.left = new TreeNode(5);

tree.root.right = new TreeNode(15);

tree.root.left.left = new TreeNode(3);

tree.root.left.right = new TreeNode(7);

int smallest = tree.findSmallest();

System.out.println("Smallest element in the binary tree: " + smallest);

}

}

**Output:-**

//Smallest element in the binary tree: 3

**// 11) Java program to find the sum of all the nodes of a binary tree:**

Program:-

class TreeNode4 {

int data;

TreeNode left;

TreeNode right;

public TreeNode4(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree4 {

TreeNode root;

public BinaryTree4() {

root = null;

}

public int findSum() {

return findSum(root);

}

private int findSum(TreeNode node) {

if (node == null) {

return 0;

}

return node.data + findSum(node.left) + findSum(node.right);

}

}

class Main5 {

public static void main(String[] args) {

BinaryTree4 tree = new BinaryTree4();

tree.root = new TreeNode(1);

tree.root.left = new TreeNode(2);

tree.root.right = new TreeNode(3);

tree.root.left.left = new TreeNode(4);

tree.root.left.right = new TreeNode(5);

int sum = tree.findSum();

System.out.println("Sum of all nodes in the binary tree: " + sum);

}

}

**Output:-**

//Sum of all nodes in the binary tree: 15

**// 12) Java program to find the total number of possible Binary Search Trees with N keys:**

**Program:-**

class Main {

public static int countBSTs(int n) {

if (n <= 1) {

return 1;

}

int count = 0;

for (int i = 1; i <= n; i++) {

count += countBSTs(i - 1) \* countBSTs(n - i);

}

return count;

}

public static void main(String[] args) {

int n = 4; // Change n to the desired number of keys.

int count = countBSTs(n);

System.out.println("Total number of possible BSTs with " + n + " keys: " + count);

}

}

**Output:-**

//Total number of possible BSTs with 4 keys: 14

**13) Java program to implement Binary Tree using the Linked List**

Program:-

class TreeNode6 {

int data;

TreeNode left;

TreeNode right;

public TreeNode6(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree6 {

TreeNode root;

public BinaryTree6() {

root = null;

}

public void insert(int data) {

root = insert(root, data);

}

private TreeNode insert(TreeNode node, int data) {

if (node == null) {

return new TreeNode(data);

}

if (data < node.data) {

node.left = insert(node.left, data);

} else if (data > node.data) {

node.right = insert(node.right, data);

}

return node;

}

public void inOrderTraversal(TreeNode node) {

if (node != null) {

inOrderTraversal(node.left);

System.out.print(node.data + " ");

inOrderTraversal(node.right);

}

}

}

class Main6 {

public static void main(String[] args) {

BinaryTree6 tree = new BinaryTree6();

tree.insert(10);

tree.insert(5);

tree.insert(15);

tree.insert(3);

tree.insert(7);

System.out.println("In-order traversal of the binary tree:");

tree.inOrderTraversal(tree.root);

}

}

**Output:-**

//In-order traversal of the binary tree:

// 3 5 7 10 15

**14) Java program to search a node in a Binary Tree**

Program:-

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

class BinaryTree {

TreeNode root;

public BinaryTree() {

root = null;

}

public boolean search(int data) {

return search(root, data);

}

private boolean search(TreeNode node, int data) {

if (node == null) {

return false;

}

if (node.data == data) {

return true;

}

boolean leftSearch = search(node.left, data);

if (leftSearch) {

return true;

}

boolean rightSearch = search(node.right, data);

return rightSearch;

}

}

public class Main {

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

tree.root = new TreeNode(10);

tree.root.left = new TreeNode(5);

tree.root.right = new TreeNode(15);

tree.root.left.left = new TreeNode(3);

tree.root.left.right = new TreeNode(7);

int searchData = 7;

boolean found = tree.search(searchData);

System.out.println("Node with data " + searchData + " found: " + found);

}

}

**Output:-**

Total number of possible BSTs with 4 keys: 14