

CSA0683-DESIGN AND ANALYSIS OF ALGORITHMS FOR SEARCHING ALGORITHMS

B.Nikhil(192111054)

DAY-1

- 1. Write a C program to Print Fibonacci Series using recursion**

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int fibonacci(int);
int main(){
    int n, i;
    printf("Enter the number of element you want in series :");
    scanf("%d",&n);
    printf("fibonacci series is : \n");
    for(i=0;i<n;i++) {
        printf("%d \n",fibonacci(i));
    }
    getch();
}
int fibonacci(int i){
    if(i==0) return 0;
    else if(i==1) return 1;
    else return (fibonacci(i-1)+fibonacci(i-2));
}
```

OUTPUT :

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program for generating a Fibonacci series. On the right, a terminal window shows the execution of the program, prompting for the number of elements and displaying the resulting series. Below the terminal is a status bar with compiler logs and system information.

```

1 #include<stdio.h>
2 #include<conio.h>
3 int fibonacci(int);
4 int main()
5 {
6     int n,i;
7     printf("Enter the number of element you want in series : ");
8     scanf("%d",&n);
9     printf("Fibonacci series is : \n");
10    for(i=0;i<n;)
11    {
12        printf("%d \n",fibonacci(i));
13    }
14    getch();
15    int fibonacci(int i){
16        if(i==0) return 0;
17        else if(i==1) return 1;
18        else return (fibonacci(i-1)+fibonacci(i-2));
19    }

```

C:\Users\VICTUS\Desktop\DA x + x

Enter the number of element you want in series :6
fibonacci series is :
1
1
2
3
5

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

Warnings: 0
Output Filename: C:\Users\VICTUS\Desktop\DA\Fibonacci Series using recursion.c
Output Size: 130.001953125 KB
Compilation Time: 0.22s

Shorten compiler paths

Line: 6 Col: 65 Sel: 0 Lines: 18 Length: 403 Insert Done parsing in 0 seconds

30°C Partly sunny Search

09:11 ENG IN 01-08-2023

2. Write a C program to check if the given no is Armstrong or not.

PROGRAM:

```
#include<stdio.h>
int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong number ");
else
printf("not armstrong number");
return 0;
}
```

OUTPUT:

```

1 #include<stdio.h>
2 int main()
3 {
4     int n,r,sum=0,temp;
5     printf("enter the number");
6     scanf("%d",&n);
7     temp=n;
8     while(n>0)
9     {
10        r=n%10;
11        sum=sum+(r*r*r);
12        n=n/10;
13    }
14    if(temp==sum)
15        printf("Armstrong number ");
16    else
17        printf("not Armstrong number");
18    return 0;
19 }

```

enter the number:153
armstrong number
Process exited after 2.785 seconds with return value 0
Press any key to continue . . . |

3. Write a C program to find the GCD of two numbers .

PROGRAM:

```

#include <stdio.h>
int main()
{
    int Num1, Num2, i, GCD;
    printf("Please Enter two integer Values \n");
    scanf("%d %d", &Num1, &Num2);
    for(i = 1; i <= Num1 && i <= Num2; i++)
    {
        if(Num1 % i == 0 && Num2 % i == 0)
            GCD = i;
    }
    printf("GCD of %d and %d is = %d", Num1, Num2, GCD);
    return 0;
}

```

OUTPUT :

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C program for calculating the Greatest Common Divisor (GCD) of two numbers. The terminal window shows the output of running the program with inputs 8 and 12, displaying the result as 4. The status bar at the bottom provides system information like temperature and date.

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int Num1, Num2, i, GCD;
6
7     printf("Please Enter two integer Values \n");
8     scanf("%d %d", &Num1, &Num2);
9
10    for(i = 1; i <= Num1 && i <= Num2; i++)
11    {
12        if(Num1 % i == 0 && Num2 % i == 0)
13        {
14            GCD = i;
15        }
16
17        printf("GCD of %d and %d is = %d", Num1, Num2, GCD);
18    }
}

```

4. Write a C program to get the largest element of an array.

PROGRAM

```

#include<stdio.h>
#include<conio.h>
int main()
{
    int arr[10], i, large;
    printf("Enter 10 Array Elements: ");
    for(i=0; i<10; i++)
        scanf("%d", &arr[i]);
    i=0;
    large = arr[i];
    while(i<10)
    {
        if(large<arr[i])
            large = arr[i];
        i++;
    }
    printf("\nLargest Number = %d", large);
    getch();
    return 0;
}

```

OUTPUT

```

1 #include<stdio.h>
2 #include<conio.h>
3
4 int main()
5 {
6     int arr[10], i, large;
7     printf("Enter 10 Array Elements: ");
8     for(i=0; i<10; i++)
9     {
10         scanf("%d", &arr[i]);
11         i=0;
12         large = arr[1];
13         while(i<10)
14         {
15             if(large<arr[i])
16                 large = arr[i];
17             i++;
18         }
19     }
20     printf("\nLargest Number = %d", large);
21     getch();
22     return 0;
23 }

```

Enter 10 Array Elements: 99
88
77
66
55
44
33
22
11
301
Largest Number = 301

5. Write a C program to find the Factorial of a number .

PROGRAM

```

#include<stdio.h>
int main()
{
    int i,fact=1,number;
    printf("Enter a number: ");
    scanf("%d",&number);
    for(i=1;i<=number;i++)
    {
        fact=fact*i;
    }
    printf("Factorial of %d is: %d",number,fact);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The top menu bar includes File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Build. The status bar at the bottom shows the current line (Line: 11), column (Col: 12), total lines (Lines: 12), total length (Length: 270), and the time taken for parsing (Done parsing in 0.015 seconds). The bottom right corner displays the date and time (09:23 01-08-2023) and system status (30°C, Partly sunny).

The code editor window displays a C program to calculate the factorial of a number:

```
1 #include<stdio.h>
2 int main()
3 {
4     int i, fact=1, number;
5     printf("Enter a number: ");
6     scanf("%d",&number);
7     for(i=1;i<=number;i++)
8     {
9         fact=fact*i;
10    }
11    printf("Factorial of %d is: %d",number,fact);
12 }
```

The terminal window shows the execution of the program:

```
C:\Users\VICIUS\Desktop\DA > Enter a number: 5
Factorial of 5 is: 120
Process exited after 1.497 seconds with return value 0
Press any key to continue . . .
```

6. Write a C program to check if a number is a prime number or not .

PROGRAM

```
#include <stdio.h>
int main()
{
    int n, i, c = 0;
    printf("Enter any number n:");
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
    {
        if (n % i == 0)
        {
            c++;
        }
    }
    if (c == 2)
    {
        printf("n is a Prime number");
    }
    else
    {
        printf("n is not a Prime number");
    }
    return 0;
}
```

OUTPUT

```

1 #include <stdio.h>
2 int main()
3 {
4     int n, i, c = 0;
5     printf("Enter any number n:");
6     scanf("%d", &n);
7     for (i = 1; i <= n; i++)
8     {
9         if (n % i == 0)
10        {
11            c++;
12        }
13    }
14    if (c == 2)
15    {
16        printf("n is a Prime number");
17    }
18    else
19    {
20        printf("n is not a Prime number");
21    }
22 }
23

```

Enter any number n:3
n is a Prime number

Process exited after 2.487 seconds with return value 0
Press any key to continue . . .

7. Write a C program to perform Selection sort.

PROGRAM

```

#include <stdio.h>
int main()
{
    int arr[10]={6,12,0,18,11,99,55,45,34,2};
    int n=10;
    int i, j, position, swap;
    for (i = 0; i < (n - 1); i++)
    {
        position = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[position] > arr[j])
                position = j;
        }
        if (position != i)
        {
            swap = arr[i];
            arr[i] = arr[position];
            arr[position] = swap;
        }
    }
    for (i = 0; i < n; i++)
        printf("%d\t", arr[i]);
    return 0;
}

```

OUTPUT

```

1 #include <stdio.h>
2 int main()
3 {
4     int arr[10]={6,12,0,18,11,99,55,45,34,2};
5     int n=10;
6     int i, j, position, swap;
7     for (i = 0; i < (n - 1); i++)
8     {
9         position = i;
10        for (j = i + 1; j < n; j++)
11        {
12            if (arr[position] > arr[j])
13                position = j;
14        }
15        if (position != i)
16        {
17            swap = arr[i];
18            arr[i] = arr[position];
19            arr[position] = swap;
20        }
21    }
22    for (i = 0; i < n; i++)
23        printf("%d\t", arr[i]);
24    return 0;
25 }

```

Process exited after 0.03447 seconds with return value 0
Press any key to continue . . . |

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

- Warnings: 0
- Output Filename: C:\Users\VICTUS\Desktop\DA\Selection Sort.c
- Output Size: 127.931640625 KB
- Compilation Time: 0.24s

Shorten compiler paths

Line: 18 Col: 33 Sel: 0 Lines: 25 Length: 518 Insert Done parsing in 0 seconds

30°C Partly sunny Search 0943 01-08-2023 ENG IN

8. Write a C program to perform Bubble sort

PROGRAM

```

#include <stdio.h>
int main()
{
    int array[100], n, c, d, swap;
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter %d integers\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    for (c = 0 ; c < n - 1; c++)
    {
        for (d = 0 ; d < n - c - 1; d++)
        {
            if (array[d] > array[d+1])
            {
                swap      = array[d];
                array[d]  = array[d+1];
                array[d+1] = swap;
            }
        }
    }
    printf("Sorted list in ascending order:\n");
}

```

```

for (c = 0; c < n; c++)
    printf("%d\n", array[c]);
return 0;
}

```

OUTPUT

```

C:\Users\VICTUS\Desktop\DAA\Bubble Sort.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TIFF-GCC 4.9.2 64-bit Release
Project Classes Debug Fibonacci Series using recursion.c Given num is armstrong or not.c GCD of Two Numbers.c find the largest element in an array.c Factorial of a number.c Given num is prime or not.c Bubble Sort.c Selection Sort.c
(globals) C:\Users\VICTUS\Desktop\DAA
1 #include <stdio.h>
2 int main()
3 {
4     int array[100], n, c, d, swap;
5     printf("Enter number of elements\n");
6     scanf("%d", &n);
7     printf("Enter %d integers\n", n);
8     for (c = 0; c < n; c++)
9         scanf("%d", &array[c]);
10    for (c = 0; c < n - 1; c++)
11    {
12        for (d = 0; d < n - c - 1; d++)
13        {
14            if (array[d] > array[d+1])
15            {
16                swap = array[d];
17                array[d] = array[d+1];
18                array[d+1] = swap;
19            }
20        }
21    }
22    printf("Sorted list in ascending order:\n");
23    for (c = 0; c < n; c++)
24        printf("%d\n", array[c]);
25    return 0;
26 }

```

Enter number of elements
6
Enter 6 integers
6
9
8
2
3
5
Sorted list in ascending order:
2
3
5
6
8
9

Process exited after 8.836 seconds with return value 0
Press any key to continue . . .

9. Write a C program to multiply two Matrix

PROGRAM

```

#include<stdio.h>
int main()
{
    int a[10][10], b[10][10], c[10][10], n, i, j, k;
    printf("Enter the value of N (N <= 10): ");
    scanf("%d", &n);
    printf("Enter the elements of Matrix-A: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Enter the elements of Matrix-B: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
    }
}

```

```
{\n    scanf("%d", & b[i][j]);\n}\n}\nfor (i = 0; i < n; i++) {\n    for (j = 0; j < n; j++) {\n        c[i][j] = 0;\n        for (k = 0; k < n; k++) {\n            c[i][j] += a[i][k] * b[k][j];\n        }\n    }\n}\nprintf("The product of the two matrices is: \n");\nfor (i = 0; i < n; i++) {\n    for (j = 0; j < n; j++) {\n        printf("%d\t", c[i][j]);\n    }\n    printf("\n");\n}\nreturn 0;\n}
```

OUTPUT

The screenshot shows the Code::Blocks IDE interface with two open files. The left file, 'Matrix Multiplication.c', contains C code for matrix multiplication using nested loops. The right file, 'DAACode.c', contains C code for calculating the product of two matrices. Both files are using the 'TDM-GCC 4.9.2 64-bit Release' compiler. The terminal window at the bottom shows the execution results for the matrix multiplication code.

```
#include<stdio.h>
int main()
{
    int a[10][10], b[10][10], c[10][10], n, i, j, k;
    printf("Enter the value of N (N <= 10): ");
    scanf("%d", &n);
    printf("Enter the elements of Matrix-A: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Enter the elements of Matrix-B: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%d", &b[i][j]);
        }
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            for (k = 0; k < n; k++)
            {
                c[i][j] = 0;
                for (k = 0; k < n; k++)
                {
                    c[i][j] += a[i][k] * b[k][j];
                }
            }
        }
    }
    printf("The product of the two matrices is: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d\t", c[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
C:\Users\VICTUS\Desktop\DAACode.c + +
Enter the value of N (N <= 10): 3
Enter the elements of Matrix-A:
1 2 3
4 5 6
7 8 9
Enter the elements of Matrix-B:
9 8 7
6 5 4
3 2 1
The product of the two matrices is:
30      24      18
84      69      54
138     114     90
-----
Process exited after 47.64 seconds with return value 0
Press any key to continue . . .
```

10. Write a C program for to check whether a given String is Palindrome or not

PROGRAM

```
#include<stdio.h>
int main()
{
    int n,r,sum=0,temp;
    printf("enter=");
    scanf("%d",&n);
    temp=n;
    while(n>0)
    {
        r=n%10;
        sum=(sum*10)+r;
        n=n/10;
    }
    if(temp==sum)
        printf("palindrome");
    else
        printf("not palindrome");
    return 0;
}
```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program named 'fibonacci Series using recursion.c'. The code checks if a given number is a palindrome. In the center, the terminal window shows the output of the program: 'enter=ANNA' followed by 'palindrome'. Below the terminal, the status bar indicates the process exited after 33.78 seconds. At the bottom, the taskbar shows the system tray with icons for battery, signal, and date/time.

```
#include<stdio.h>
int main()
{
    int n,r,sum=0,temp;
    printf("enter=");
    scanf("%d",&n);
    temp=n;
    while(n>0)
    {
        r=n%10;
        sum=(sum*10)+r;
        n=n/10;
    }
    if(temp==sum)
        printf("palindrome");
    else
        printf("not palindrome");
    return 0;
}
```

11. Write a C program to copy one string to another

PROGRAM

```
#include<stdio.h>
```

```

#include<conio.h>
#include<string.h>
int main()
{
    char str1[20], str2[20];
    printf("Enter the string: ");
    gets(str1);
    printf("\nString 1 = %s", str1);
    strcpy(str2, str1);
    printf("\nString 2 = %s", str2);
    getch();
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The code in the editor is identical to the one above. In the terminal window, the user enters "NIKHIL" and the program outputs "String 1 = NIKHIL" and "String 2 = NIKHIL". The status bar at the bottom shows the date and time as 01-08-2023 10:24.

```

1 #include<stdio.h>
2 #include<conio.h>
3 #include<string.h>
4 int main()
5 {
6     char str1[20], str2[20];
7     printf("Enter the string: ");
8     gets(str1);
9     printf("\nString 1 = %s", str1);
10    strcpy(str2, str1);
11    printf("\nString 2 = %s", str2);
12    getch();
13    return 0;
14 }

```

12. Write a Program to perform binary search.

PROGRAM

```

#include <stdio.h>
int main()
{
    int c, first, last, middle, n, search, array[100];
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter %d integers\n", n);
    for (c = 0; c < n; c++)

```

```

scanf("%d", &array[c]);
printf("Enter value to find\n");
scanf("%d", &search);
first = 0;
last = n - 1;
middle = (first+last)/2;
while (first <= last)
{
    if (array[middle] < search)
        first = middle + 1;
    else if (array[middle] == search)
    {
        printf("%d found at location %d.\n", search, middle+1);
        break;
    }
    else
        last = middle - 1;
    middle = (first + last)/2;
}
if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);
return 0;
}

```

OUTPUT

```

C:\Users\VICTUS\Desktop\DA\Binary Search.c [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
File Project View Execute Tools AStyle Window Help
(globals) Fibonacci Series using recursion.c Given num is armstrong or not.c GCD of Two Numbers.c find the largest element in an array.c Factorial of a number.c Given num is prime or not.c Bubble Sort.c
Project Classes Debug Fibonacci Series using recursion.c Given num is armstrong or not.c GCD of Two Numbers.c find the largest element in an array.c Factorial of a number.c Given num is prime or not.c Bubble Sort.c
#include <stdio.h>
int main()
{
    int c, first, last, middle, n, search, array[100];
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter %d integers\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    printf("Enter value to find\n");
    scanf("%d", &search);
    first = 0;
    last = n - 1;
    middle = (first+last)/2;
    while (first <= last)
    {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search)
        {
            printf("%d found at location %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;
        middle = (first + last)/2;
    }
    if (first > last)
        printf("Not found! %d isn't present in the list.\n", search);
    return 0;
}

C:\Users\VICTUS\Desktop\DA [ ] + -
Enter number of elements
3
Enter 3 integers
55
66
88
Enter value to find
66
66 found at location 2.

Process exited after 9.968 seconds with return value 0
Press any key to continue . . .

```

Line: 19 Col: 39 Sel: 0 Lines: 31 Length: 759 Insert Done parsing in 0.016 seconds

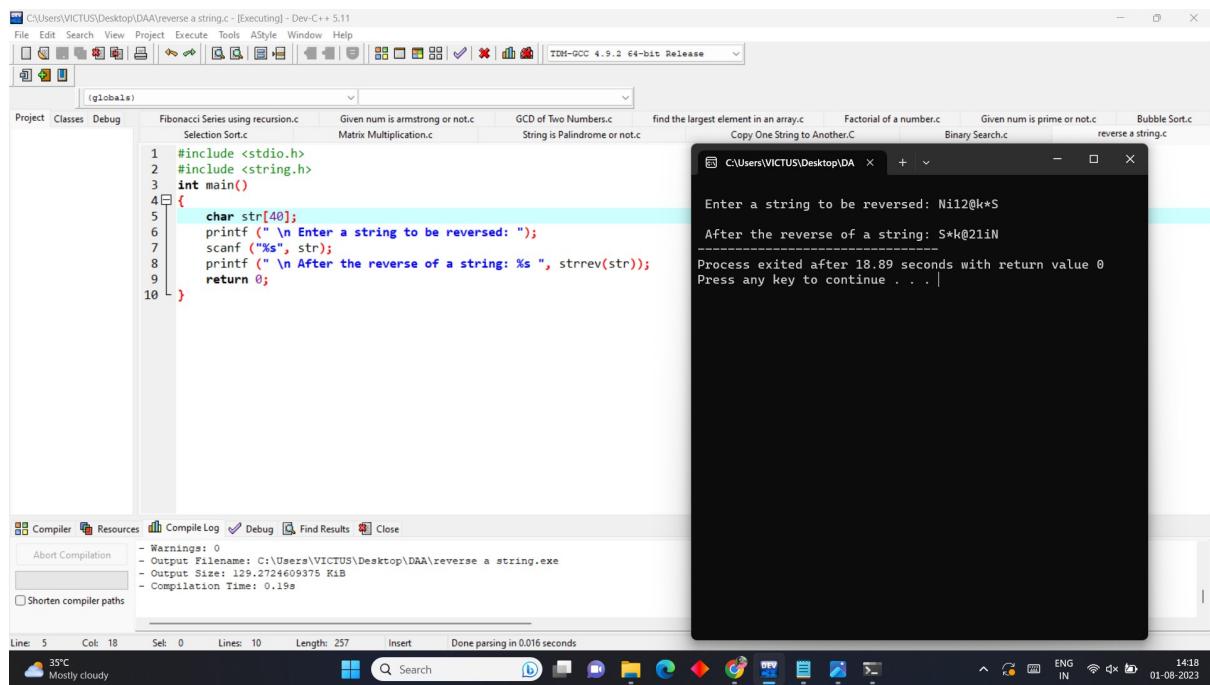
32°C Mostly cloudy Search 10:40 01-08-2023

13. Write a program to print the reverse of a string

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[40];
    printf (" \n Enter a string to be reversed: ");
    scanf ("%s", str);
    printf (" \n After the reverse of a string: %s ", strrev(str));
    return 0;
}
```

OUTPUT



14. Write a program to find the length of a string.

PROGRAM

```
#include <stdio.h>
int main() {
    char s[] = "Good Morning";
    int i;

    for (i = 0; s[i] != '\0'; ++i);

    printf("Length of the string: %d", i);
    return 0;
}
```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The main window displays a C program to calculate the length of a string. The code uses a for loop to iterate through each character of the string until it finds a null terminator ('\\0'). The output window shows the string "Good Morning" and its length, 12. The status bar at the bottom provides system information like temperature and date.

```
#include <stdio.h>
int main()
{
    char s[] = "Good Morning";
    int i;
    for (i = 0; s[i] != '\0'; ++i);
    printf("Length of the string: %d", i);
    return 0;
}
```

Length of the string: 12
Process exited after 0.05861 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation - Warning(s) 0 - Output Filename: C:\Users\VIKTUS\Desktop\DA\Find the length of string.c - Output Size: 127.955984375 KiB - Compilation Time: 0.20s
Shorten compiler paths

Line: 4 Col: 22 Sel: 0 Lines: 10 Length: 181 Insert Done parsing

35°C Mostly cloudy 14:28 01-08-2023 ENG IN

15. Write a program to perform Strassen's Matrix Multiplication.

PROGRAM

```
#include<stdio.h>
int main(){
    int a[2][2], b[2][2], c[2][2], i, j;
    int m1, m2, m3, m4 , m5, m6, m7;

    printf("Enter the 4 elements of first matrix: ");
    for(i = 0;i < 2; i++)
        for(j = 0;j < 2; j++)
            scanf("%d", &a[i][j]);

    printf("Enter the 4 elements of second matrix: ");
    for(i = 0; i < 2; i++)
        for(j = 0;j < 2; j++)
            scanf("%d", &b[i][j]);

    printf("\nThe first matrix is\n");
    for(i = 0; i < 2; i++){
        printf("\n");
        for(j = 0; j < 2; j++)
            printf("%d\t", a[i][j]);
    }

    printf("\nThe second matrix is\n");
```

```

for(i = 0;i < 2; i++){
    printf("\n");
    for(j = 0;j < 2; j++)
        printf("%d\t", b[i][j]);
}

m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
m2= (a[1][0] + a[1][1]) * b[0][0];
m3= a[0][0] * (b[0][1] - b[1][1]);
m4= a[1][1] * (b[1][0] - b[0][0]);
m5= (a[0][0] + a[0][1]) * b[1][1];
m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]);
m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]);

c[0][0] = m1 + m4- m5 + m7;
c[0][1] = m3 + m5;
c[1][0] = m2 + m4;
c[1][1] = m1 - m2 + m3 + m6;

printf("\nAfter multiplication using Strassen's algorithm \n");
for(i = 0; i < 2 ; i++){
    printf("\n");
    for(j = 0;j < 2; j++)
        printf("%d\t", c[i][j]);
}

return 0;
}

```

OUTPUT

C:\Users\VICTUS\Desktop\DA\Strassen Matrix Multiplication.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

Project Classes Debug Fibonacci Series using recursion.c Given num is armstrong or not.c GCD of Two Numbers.c Selection Sort.c Matrix Multiplication.c String is Palindrome or not.c Copy One String.c

(globals)

```

1  #include<stdio.h>
2
3  int main()
4  {
5      int matrix[4][4], a11[2][2], a12[2][2], a21[2][2], a22[2][2];
6      int m1, m2, m3, m4, m5, m6, m7, m8;
7
8      printf("Enter the 4 elements of first matrix: ");
9      for(i = 0; i < 2; i++)
10     {
11         for(j = 0; j < 2; j++)
12         {
13             scanf("%d", &a11[i][j]);
14         }
15     }
16
17     printf("Enter the 4 elements of second matrix: ");
18     for(i = 0; i < 2; i++)
19     {
20         for(j = 0; j < 2; j++)
21         {
22             scanf("%d", &a21[i][j]);
23         }
24     }
25
26     printf("The first matrix is\n");
27     for(i = 0; i < 2; i++)
28     {
29         for(j = 0; j < 2; j++)
30         {
31             printf("%d\t", a11[i][j]);
32         }
33         printf("\n");
34     }
35
36     printf("The second matrix is\n");
37     for(i = 0; i < 2; i++)
38     {
39         for(j = 0; j < 2; j++)
40         {
41             printf("%d\t", a21[i][j]);
42         }
43         printf("\n");
44     }
45
46     m1 = (a11[0][0] * a12[0][0]) + (a11[0][1] * a12[1][0]);
47     m2 = (a11[0][0] * a12[0][1]) + (a11[0][1] * a12[1][1]);
48     m3 = (a11[1][0] * a12[0][0]) + (a11[1][1] * a12[1][0]);
49     m4 = (a11[1][0] * a12[0][1]) + (a11[1][1] * a12[1][1]);
50     m5 = (a21[0][0] * a22[0][0]) + (a21[0][1] * a22[1][0]);
51     m6 = (a21[0][0] * a22[0][1]) + (a21[0][1] * a22[1][1]);
52     m7 = (a21[1][0] * a22[0][0]) + (a21[1][1] * a22[1][0]);
53     m8 = (a21[1][0] * a22[0][1]) + (a21[1][1] * a22[1][1]);
54
55     printf("%d\t %d\t %d\t %d\n", m1, m2, m3, m4);
56     printf("%d\t %d\t %d\t %d\n", m5, m6, m7, m8);
57     printf("%d\t %d\t %d\t %d\n", m3, m4, m1, m2);
58     printf("%d\t %d\t %d\t %d\n", m7, m8, m5, m6);
59
60     printf("After multiplication using Strassen's algorithm\n");
61
62     for(i = 0; i < 2; i++)
63     {
64         for(j = 0; j < 2; j++)
65         {
66             printf("%d\t", a11[i][j]);
67         }
68         printf("\n");
69     }
70
71     return 0;
72 }
```

Enter the 4 elements of first matrix:
1 2
3 4
Enter the 4 elements of second matrix:
5 6
7 8
The first matrix is
1 2
3 4
The second matrix is
5 6
7 8
After multiplication using Strassen's algorithm
19 22
43 50

Process exited after 13.1 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results

Line: 20 Col: 36 Sel: 0 Lines: 51 Length: 1345 Insert Done parsing in 0.015 seconds

Cloud 35°C Mostly cloudy

Search ENG IN 14:55 01-08-2023

CSA0683-DESIGN AND ANALYSIS OF ALGORITHMS FOR SEARCHING ALGORITHMS

B.Nikhil(192111054)

DAY-2

- 1. Write a program to perform MST using greedy techniques.**

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10] = {
    0
}
,min,mincost=0,cost[10][10];
void main() {
    clrscr();
    printf("\n Enter the number of nodes:");
    scanf("%d",&n);
    printf("\n Enter the adjacency matrix:\n");
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    visited[1]=1;
    printf("\n");
    while(ne<n) {
        for (i=1,min=999;i<=n;i++)
            for (j=1;j<=n;j++)
                if(cost[i][j]<min)
                    if(visited[i]!=0) {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
        if(visited[u]==0 || visited[v]==0) {
            printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
            mincost+=min;
            visited[b]=1;
        }
        cost[a][b]=cost[b][a]=999;
    }
}
```

```

        printf("\n Minimum cost=%d",mincost);
        getch();
    }
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code for 'MST.c' is displayed:

```

1 #include<stdio.h>
2 #include<conio.h>
3 int a,b,u,v,i,j,n,e=1;
4 int visited[10]={0};
5
6
7 ,min,mincost=0,cost[10][10];
8 int main() {
9     printf("\nEnter the number of nodes:");
10    scanf("%d",&n);
11    printf("\nEnter the adjacency matrix:\n");
12    for (i=1;i<n;i++) {
13        for (j=1;j<n;j++) {
14            scanf("%d",&cost[i][j]);
15            if(cost[i][j]==0)
16                cost[i][j]=999;
17        }
18        visited[i]=1;
19        printf("\n");
20    }
21    while(ne>n) {
22        for (i=1;i<n;i++)
23            for (j=i;j<n;j++)
24                if(cost[i][j]<min) {
25                    if(visited[i]==0) {
26                        min=cost[i][j];
27                        a=u;
28                        b=v;
29                    }
30                    if(visited[u]==0 || visited[v]==0) {
31                        printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
32                        mincost+=min;
33                        visited[b]=1;
34                    }
35                    cost[a][b]-=cost[b][a]=999;
36                }
37    }
38    printf("\n Minimum cost=%d",mincost);
39    getch();
}

```

On the right, a terminal window shows the program's output:

```

Enter the number of nodes:6
Enter the adjacency matrix:
0 5 6 4 0 0
5 0 1 2 0 0
6 1 0 2 5 3
4 2 2 0 0 4
0 0 5 0 0 4
0 0 3 4 4 0

Edge 1:(1 4) cost:4
Edge 2:(4 2) cost:2
Edge 3:(2 3) cost:1
Edge 4:(3 6) cost:3
Edge 5:(6 5) cost:4
Minimum cost=14

```

2. Using Dynamic programming concepts to find out the Optimal binary search tree

PROGRAM

```

#include <stdio.h>
#include <limits.h>

double sumProbabilities(double probabilities[], int start, int end) {
    double sum = 0.0;
    for (int i = start; i <= end; i++)
        sum += probabilities[i];
    return sum;
}

double optimalBSTCost(double keys[], double probabilities[], int n) {
    double dp[n][n];
    for (int i = 0; i < n; i++)
        dp[i][i] = probabilities[i];
    for (int L = 2; L <= n; L++) {
        for (int i = 0; i <= n - L + 1; i++) {
            int j = i + L - 1;
            dp[i][j] = INT_MAX;
            for (int r = i; r <= j; r++) {
                double cost = ((r > i) ? dp[i][r - 1] : 0) + ((r < j) ? dp[r + 1][j] : 0) +
sumProbabilities(probabilities, i, j);
                if (cost < dp[i][j])
                    dp[i][j] = cost;
            }
        }
    }
}

```

```

        }
    }

    return dp[0][n - 1];
}

int main()
{
    int n;
    printf("Enter the number of keys: ");
    scanf("%d", &n);
    double keys[n], probabilities[n];
    printf("Enter the keys and their probabilities:\n");
    for (int i = 0; i < n; i++) {
        scanf("%lf %lf", &keys[i], &probabilities[i]);
    }
    double cost = optimalBSTCost(keys, probabilities, n);
    printf("Optimal BST cost: %lf\n", cost);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left is the code editor with the file 'MST.c' open, containing the C program for calculating the optimal BST cost. On the right is a terminal window showing the execution of the program. The terminal output is as follows:

```

C:\Users\VIKTUS\Desktop\Optimal BST.C - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools Style Window Help
[global]
Project Classes Debug MST.c [*] Optimal BST.C Binomial Coefficient.cpp
1 #include <stdio.h>
2 #include <limits.h>
3 double sumProbabilities(double probabilities[], int start, int end) {
4     double sum = 0.0;
5     for (int i = start; i <= end; i++)
6         sum += probabilities[i];
7     return sum;
8 }
9 double optimalBSTCost(double keys[], double probabilities[], int n) {
10    double dp[n][n];
11    for (int i = 0; i < n; i++)
12        dp[i][i] = probabilities[i];
13    for (int l = 1; l < n; l++) {
14        for (int i = 0; i <= n - l - 1; i++) {
15            int j = i + l - 1;
16            dp[i][j] = INT_MAX;
17            for (int r = i; r <= j; r++) {
18                double cost = ((r > i) ? dp[i][r - 1] : 0) + ((r < j) ? dp[r + 1][j] : 0) + sumProbabilities(probabilities, i, j);
19                if (cost < dp[i][j])
20                    dp[i][j] = cost;
21            }
22        }
23    }
24    return dp[0][n - 1];
25 }
26 int main()
27 {
28     int n;
29     printf("Enter the number of keys: ");
30     scanf("%d", &n);
31     double keys[n];
32     printf("Enter the keys and their probabilities:\n");
33     for (int i = 0; i < n; i++) {
34         scanf("%lf %lf", &keys[i], &probabilities[i]);
35     }
36     double cost = optimalBSTCost(keys, probabilities, n);
37     printf("Optimal BST cost: %lf\n", cost);
38     return 0;
39 }

```

Enter the number of keys: 4
Enter the keys and their probabilities:
10 15
20 24
30 10
40 5
Optimal BST cost: 89.000000

Process exited after 27.36 seconds with return value 0
Press any key to continue . . .

3. Using Dynamic programming techniques to find the binomial coefficient of a given number

PROGRAM

```

#include <stdio.h>
unsigned long long binomialCoefficient(int n, int r) {
    unsigned long long dp[n + 1][r + 1];
    for (int i = 0; i <= n; i++) {

```

```

for (int j = 0; j <= r && j <= i; j++) {
    if (j == 0 || j == i)
        dp[i][j] = 1;
    else
        dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
}
}
return dp[n][r];
}

int main() {
    int n, r;
    printf("Enter the value of n : ");
    scanf("%d", &n);
    printf("Enter the value of r : ");
    scanf("%d", &r);
    unsigned long long result = binomialCoefficient(n, r);
    printf("Binomial Coefficient C(%d, %d) = %llu\n", n, r, result);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code for 'Binomial Coefficient.cpp' is displayed. It includes a function to calculate the binomial coefficient using dynamic programming and a main function to take user input and print the result. On the right, a terminal window shows the execution of the program. It prompts for values of n and r, calculates the result (C(7, 5) = 21), and then exits.

```

C:\Users\Victus\Desktop\DA\Binomial Coefficient.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools Style Window Help
File Project Classes Debug MST.c Optimal BST.C Binomial Coefficient.cpp
(globals)
Project Classes Debug MST.c Optimal BST.C Binomial Coefficient.cpp
1 #include <stdio.h>
2 unsigned long long binomialCoefficient(int n, int r) {
3     unsigned long long dp[n + 1][r + 1];
4     for (int i = 0; i <= n; i++) {
5         for (int j = 0; j <= r && j <= i; j++) {
6             if (j == 0 || j == i)
7                 dp[i][j] = 1;
8             else
9                 dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
10        }
11    }
12    return dp[n][r];
13 }
14 int main() {
15     int n, r;
16     printf("Enter the value of n : ");
17     scanf("%d", &n);
18     printf("Enter the value of r : ");
19     scanf("%d", &r);
20     unsigned long long result = binomialCoefficient(n, r);
21     printf("Binomial Coefficient C(%d, %d) = %llu\n", n, r, result);
22     return 0;
23 }

C:\Users\Victus\Desktop\DA x
Enter the value of n : 7
Enter the value of r : 5
Binomial Coefficient C(7, 5) = 21
Process exited after 5.475 seconds with return value 0
Press any key to continue . . .

```

4. Write a program to perform the Knapsack problem using greedy techniques

PROGRAM

```

#include <stdio.h>
#include <stdlib.h>
struct Item {
    int weight;

```

```

int value;
};

int compare(const void* a, const void* b) {
    struct Item* itemA = (struct Item*)a;
    struct Item* itemB = (struct Item*)b;
    double valuePerUnitWeightA = (double)itemA->value / itemA->weight;
    double valuePerUnitWeightB = (double)itemB->value / itemB->weight;
    if (valuePerUnitWeightA < valuePerUnitWeightB)
        return 1;
    else if (valuePerUnitWeightA > valuePerUnitWeightB)
        return -1;
    else
        return 0;
}

double fractionalKnapsack(int capacity, struct Item items[], int n) {
    qsort(items, n, sizeof(struct Item), compare);
    double totalValue = 0.0;
    int currentWeight = 0;
    for (int i = 0; i < n; i++) {
        if (currentWeight + items[i].weight <= capacity) {
            currentWeight += items[i].weight;
            totalValue += items[i].value;
        } else {
            int remainingWeight = capacity - currentWeight;
            totalValue += items[i].value * ((double)remainingWeight / items[i].weight);
            break;
        }
    }
    return totalValue;
}

int main() {
    int capacity = 50;
    struct Item items[] = { {10, 60}, {20, 100}, {30, 120} };
    int n = sizeof(items) / sizeof(items[0]);
    double maxValue = fractionalKnapsack(capacity, items, n);
    printf("Maximum value in the knapsack: %.2f\n", maxValue);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays `knapsack.cpp` with C++ code for solving a knapsack problem using dynamic programming. On the right, a terminal window shows the execution results: "Maximum value in the knapsack: 240.00" and "Process exited after 0.05638 seconds with return value 0". Below the IDE is a Windows taskbar with various icons and system status.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Item {
4     int weight;
5     int value;
6 };
7 int compare(const void* a, const void* b) {
8     struct Item* itemA = (struct Item*)a;
9     struct Item* itemB = (struct Item*)b;
10    double valuePerUnitWeightA = (double)itemA->value / itemA->weight;
11    double valuePerUnitWeightB = (double)itemB->value / itemB->weight;
12    if (valuePerUnitWeightA < valuePerUnitWeightB)
13        return 1;
14    else if (valuePerUnitWeightA > valuePerUnitWeightB)
15        return -1;
16    else
17        return 0;
18 }
19 double fractionalKnapsack(int capacity, struct Item items[], int n) {
20     qsort(items, n, sizeof(struct Item), compare);
21     double totalValue = 0.0;
22     int currentWeight = 0;
23     for (int i = 0; i < n; i++) {
24         if (currentWeight + items[i].weight <= capacity) {
25             currentWeight += items[i].weight;
26             totalValue += items[i].value;
27         } else {
28             int remainingWeight = capacity - currentWeight;
29             totalValue += items[i].value * ((double)remainingWeight / items[i].weight);
30             break;
31         }
32     }
33     return totalValue;
34 }
35 int main() {
36     int capacity = 50;
37     struct Item items[] = { {10, 60}, {20, 100}, {30, 120} };
38     int n = sizeof(items) / sizeof(items[0]);
39     double maxValue = fractionalKnapsack(capacity, items, n);
40     printf("Maximum value in the knapsack: %.2f\n", maxValue);
41     return 0;
42 }

```

5. Write a program to generate all the prime numbers.

PROGRAM

```

#include <stdio.h>
int main()
{
    int i, j, end, isPrime;
    printf("Find prime numbers between 1 to : ");
    scanf("%d", &end);
    printf("All prime numbers between 1 to %d are:\n", end);
    for(i=2; i<=end; i++)
    {
        isPrime = 1;
        for(j=2; j<=i/2; j++)
        {
            if(i%j==0)
            {
                isPrime = 0;
                break;
            }
        }
        if(isPrime==1)
        {
            printf("%d, ", i);
        }
    }

    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C program named 'Generate all the prime numbers.c' which finds prime numbers between 1 and a user-specified number n. The output window shows the prime numbers from 2 to 7. The status bar at the bottom provides system information like weather and date.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, j, end, isPrime;
6     printf("Find prime numbers between 1 to : ");
7     scanf("%d", &end);
8     printf("All prime numbers between 1 to %d are:\n", end);
9     for(i=2; i<end; i++)
10    {
11        isPrime = 1;
12        for(j=2; j<=i/2; j++)
13        {
14            if((i*j)==0)
15            {
16                isPrime = 0;
17                break;
18            }
19        }
20        if(isPrime==1)
21        {
22            printf("%d ", i);
23        }
24    }
25
26    return 0;
27 }
```

C:\Users\VICTUS\Desktop\DA x + v
Find prime numbers between 1 to : 9
All prime numbers between 1 to 9 are:
2, 3, 5, 7,

Process exited after 3.93 seconds with return value 0
Press any key to continue . . . |

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 26 Col: 2 Sel: 0 Lines: 26 Length: 519 Insert Done parsing in 0.016 seconds
35°C Mostly cloudy Search 13:49 ENG IN 02-08-2023

6.Using Divide and Conquer strategy to find Max and Min values in the list.

PROGRAM

```
#include<stdio.h>
#include<stdio.h>
int max, min;
int a[100];
void maxmin(int i, int j)
{
int max1, min1, mid;
if(i==j)
{
max = min = a[i];
}
else
{
if(i == j-1)
{
if(a[i] <a[j])
{
max = a[j];
min = a[i];
}
else
{
max = a[i];
min = a[j];
}
}
else
{
max = a[i];
min = a[j];
}
```

```

    }
}
else
{
    mid = (i+j)/2;
    maxmin(i, mid);
    max1 = max; min1 = min;
    maxmin(mid+1, j);
    if(max <max1)
        max = max1;
    if(min > min1)
        min = min1;
}
}
}
int main ()
{
    int i, num;
    printf ("\nEnter the total number of numbers : ");
    scanf ("%d",&num);
    printf ("Enter the numbers : \n");
    for (i=1;i<=num;i++)
        scanf ("%d",&a[i]);
    max = a[0];
    min = a[0];
    maxmin(1, num);
    printf ("Minimum element in an array : %d\n", min);
    printf ("Maximum element in an array : %d\n", max);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The project navigation bar at the top lists several files: Optimal BST.C, Binomial Coefficient.cpp, MST.c, knapsack.cpp, Generate all the prime numbers.c, Merge Sort.c, Divide and Conquer for max and min.c. The main code editor window displays the C code for finding the min and max of an array. To the right of the editor is a terminal window titled 'C:\Users\VICUS\Desktop\DA' showing the execution of the program. The terminal output is as follows:

```

Enter the total number of numbers : 5
Enter the numbers :
20
30
40
50
99
Minimum element in an array : 20
Maximum element in an array : 99
-----
```

Below the terminal window, the status bar indicates the process exited after 7.512 seconds with a return value of 0, and it prompts the user to press any key to continue.

7. Write a program to perform Merge Sort.

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
```

```

        merge(arr, l, m, r);
    }

}

void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is :\n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface with the following details:

- Project Bar:** C:\Users\VIKTUS\Desktop\DAAMerge Sort.c - [Executing] - Dev-C++ 5.11
- Toolbar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help
- Compiler Bar:** TDM-GCC 4.9.2 64-bit Release
- Code Editor:** Shows the C code for merge sort.
- Terminal Window:**
 - Output: Given array is :
12 11 13 5 6 7
 - Output: Sorted array is
5 6 7 11 12 13
 - Output: Process exited after 0.04266 seconds with return value 0
 - Message: Press any key to continue . . .
- Status Bar:** Line: 43 Col: 30 Sel: 0 Lines: 66 Length: 1120 Insert Done parsing in 0 seconds
- System Tray:** 36°C Mostly cloudy
- System Bar:** ENG IN 14:52 02-08-2023

CSA0683-DESIGN AND ANALYSIS OF ALGORITHMS FOR SEARCHING ALGORITHMS

B.Nikhil(192111054)

DAY-3

1. Write C program to perform n Queens Problem using backtracking

PROGRAM

```
#include<stdio.h>
#include<math.h>
int board[20],count;
int main()
{
    int n,i,j;
    void queen(int row,int n);
    printf(" - N Queens Problem Using Backtracking -");
    printf("\n\nEnter number of Queens:");
    scanf("%d",&n);
    queen(1,n);
    return 0;
}
void print(int n)
{
    int i,j;
    printf("\n\nSolution %d:\n\n",++count);
    for(i=1;i<=n;++i)
        printf("\t%d",i);
    for(i=1;i<=n;++i)
    {
        printf("\n\n%d",i);
        for(j=1;j<=n;++j)
        {
            if(board[i]==j)
                printf("\tQ");
            else
                printf("\t-");
        }
    }
}
int place(int row,int column)
{
    int i;
```

```

for(i=1;i<=row-1;++i)
{
    if(board[i]==column)
        return 0;
    else
        if(abs(board[i]-column)==abs(i-row))
            return 0;
}
return 1;
}
void queen(int row,int n)
{
int column;
for(column=1;column<=n;++column)
{
if(place(row,column))
{
    board[row]=column;
    if(row==n)
        print(n);
    else
        queen(row+1,n);
}
}
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface with the code for the N-Queens problem. The user has entered '4' as the number of queens. The IDE displays two solutions:

Solution 1:

	1	2	3	4
1	-	Q	-	-
2	-	-	-	Q
3	Q	-	-	-
4	-	-	Q	-

Solution 2:

	1	2	3	4
1	-	-	Q	-
2	Q	-	-	-
3	-	-	-	Q
4	-	Q	-	-

Process exited after 3.363 seconds with return value 0
Press any key to continue . . .

2. Write C program to print minimum and maximum value sequence for all the numbers in a list.

PROGRAM

```
#include <stdio.h>
void main()
{
    int arr1[100];
    int i, mx, mn, n;
    printf("Input the number of elements to be stored in the array :");
    scanf("%d",&n);
    printf("Input %d elements in the array :\n",n);
    for(i=0;i<n;i++)
    {
        printf("element - %d : ",i);
        scanf("%d",&arr1[i]);
    }
    mx = arr1[0];
    mn = arr1[0];
    for(i=1; i<n; i++)
    {
        if(arr1[i]>mx)
        {
            mx = arr1[i];
        }
        if(arr1[i]<mn)
        {
            mn = arr1[i];
        }
    }
    printf("Maximum element is : %d\n", mx);
    printf("Minimum element is : %d\n\n", mn);
}
```

OUTPUT

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\VICTUS\Desktop\DA\max and min in an array.c - [Executing] - Dev-C++ 5.11
- Menu Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbar:** Includes icons for New, Open, Save, Print, Cut, Copy, Paste, Find, Replace, Undo, Redo, etc.
- Project Explorer:** Shows the file "n-queenc.c" under the "Project" section.
- Code Editor:** Displays the C code for finding the maximum and minimum elements in an array. The code includes input validation for the number of elements and the array values.
- Output Window:** Shows the terminal output of the program execution. It prompts for the number of elements (Input 6 elements in the array :), lists the array elements (element - 0 : 20, element - 1 : 30, etc.), and prints the maximum and minimum values (Maximum element is : 700, Minimum element is : -90).
- Status Bar:** Shows compiler information (TDM-GCC 4.9.2 64-bit Release), line count (Line: 5), column count (Col: 22), selection count (Sel: 0), lines (Lines: 29), length (Length: 670), and parsing time (Done parsing in 0.031 seconds).
- Taskbar:** Shows the Start button, Search bar, and various application icons.

3. Write C program to perform traveling salesman problem using dynamic programming

PROGRAM

```
#include<stdio.h>
int ary[10][10],completed[10],n, cost=0;
void takeInput()
{
    int i,j;
    printf("Enter the number of villages: ");
    scanf("%d",&n);
    printf("\nEnter the Cost Matrix\n");
    for(i=0;i < n;i++)
    {
        printf("\nEnter Elements of Row: %d\n",i+1);
        for( j=0;j < n;j++)
            scanf("%d",&ary[i][j]);
        completed[i]=0;
    }
    printf("\n\nThe cost list is:");
    for( i=0;i < n;i++)
    {
        printf("\n");
        for(j=0;j < n;j++)
            printf("\t%d",ary[i][j]);
    }
}
```

```

}

void mincost(int city)
{
int i,ncity;
completed[city]=1;
printf("%d--->",city+1);
ncity=least(city);
if(ncity==999)
{
ncity=0;
printf("%d",ncity+1);
cost+=ary[city][ncity];
return;
}
mincost(ncity);
}

int least(int c)
{
int i,nc=999;
int min=999,kmin;
for(i=0;i < n;i++)
{
if((ary[c][i]!=0)&&(completed[i]==0))
if(ary[c][i]+ary[i][c] < min)
{
min=ary[i][0]+ary[c][i];
kmin=ary[c][i];
nc=i;
}
}
if(min!=999)
cost+=kmin;
return nc;
}
int main()
{
takeInput();
printf("\n\nThe Path is:\n");
mincost(0);
printf("\n\nMinimum cost is %d\n ",cost);
return 0;
}

```

OUTPUT

```

1 #include<stdio.h>
2 int ary[10][10],completed[10],n,cost=0;
3 void takeInput()
4 {
5     int i,j;
6     printf("Enter the number of villages: ");
7     scanf("%d",&n);
8     printf("\nEnter the Cost Matrix\n");
9     for(i=0;i < n;i++)
10    {
11        printf("\nEnter Elements of Row: %d\n",i+1);
12        for( j=0;j < n;j++)
13            scanf("%d",&ary[i][j]);
14        completed[i]=0;
15    }
16    printf("\n\nThe cost list is:");
17    for( i=0;i < n;i++)
18    {
19        printf("\n");
20        for(j=0;j < n;j++)
21            printf("%d",ary[i][j]);
22    }
23 }
24 void mincost(int city)
25 {
26     int i,ncity;
27     completed[city]=1;
28     printf("%d---",city+1);
29     ncity=least(city);
30     if(ncity==999)
31    {
32         ncity=0;
33         printf("%d",ncity+1);
34         cost+=ary[city][ncity];

```

Enter the number of villages: 3
Enter the Cost Matrix
Enter Elements of Row: 1 2 3
Enter Elements of Row: 2 4 5 6
Enter Elements of Row: 3 7 8 9
The cost list is:
1 2 3
4 5 6
7 8 9
The Path is:
1--->2--->3--->1
Minimum cost is 15

Process exited after 11.49 seconds with return value 0
Press any key to continue . . . |

4. Write C program for the given pattern

If n=4

```

1
1 2
1 2 3
1 2 3 4

```

PROGRAM:

```

#include <stdio.h>
int main()
{
    int i, j, rows;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    for (i = 1; i <= rows; i++)
    {
        for (j = 1; j <= i; j++)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program for printing a pattern of numbers. The code uses nested loops to print rows of numbers from 1 to 4. On the right, a terminal window titled 'C:\Users\...\\DA' shows the execution of the program. It prompts the user for the number of rows (4), prints the pattern, and then exits. The taskbar at the bottom shows the system status, including the date and time.

```

1 #include <stdio.h>
2 int main()
3 {
4     int i, j, rows;
5     printf("Enter the number of rows: ");
6     scanf("%d", &rows);
7     for (i = 1; i <= rows;i++)
8     {
9         for (j = 1; j <= i;j++)
10        {
11            printf("%d ", j);
12        }
13        printf("\n");
14    }
15    return 0;
16 }
17

```

5. Write C program to perform Floyd's algorithm

PROGRAM

```

#include <stdio.h>
#include <stdlib.h>
void floydWarshall(int **graph, int n)
{
    int i, j, k;
    for (k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (graph[i][j] > graph[i][k] + graph[k][j])
                    graph[i][j] = graph[i][k] + graph[k][j];
            }
        }
    }
}
int main(void)
{
    int n, i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    int **graph = (int **)malloc((long unsigned) n * sizeof(int *));

```

```

for (i = 0; i < n; i++)
{
    graph[i] = (int *)malloc((long unsigned) n * sizeof(int));
}
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (i == j)
            graph[i][j] = 0;
        else
            graph[i][j] = 100;
    }
}
printf("Enter the edges: \n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        printf("[%d][%d]: ", i, j);
        scanf("%d", &graph[i][j]);
    }
}
printf("The original graph is:\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        printf("%d ", graph[i][j]);
    }
    printf("\n");
}
floydWarshall(graph, n);
printf("The shortest path matrix is:\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        printf("%d ", graph[i][j]);
    }
    printf("\n");
}
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program for the Floyd-Warshall algorithm. On the right, the terminal window shows the execution of the program. The user enters the number of vertices as 3. The program then asks for edges and processes them to find the shortest path matrix. The original graph and the resulting shortest path matrix are printed. The process exits after 22.57 seconds.

```
#include <stdio.h>
#include <stdlib.h>
void floydWarshall(int **graph, int n)
{
    int i, j, k;
    for (k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (graph[i][j] > graph[i][k] + graph[k][j])
                    graph[i][j] = graph[i][k] + graph[k][j];
            }
        }
    }
}

int main(void)
{
    int n, i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    int **graph = (int **)malloc((long unsigned) n * sizeof(int *));
    for (i = 0; i < n; i++)
    {
        graph[i] = (int *)malloc((long unsigned) n * sizeof(int));
        for (j = 0; j < n; j++)
        {
            if (i == j)
                graph[i][j] = 0;
            else
                graph[i][j] = 100;
        }
    }
    printf("Enter the edges: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
    }
}

Process exited after 22.57 seconds with return value 0
Press any key to continue . . .
```

6. Write C program for pascal triangle.

PROGRAM

```
#include <stdio.h>
int main() {
    int rows, coef = 1, space, i, j;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    for (i = 0; i < rows; i++) {
        for (space = 1; space <= rows - i; space++)
            printf(" ");
        for (j = 0; j <= i; j++) {
            if (j == 0 || i == 0)
                coef = 1;
            else
                coef = coef * (i - j + 1) / j;
            printf("%4d", coef);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT

C:\Users\VIKTUS\Desktop\DA\pascal triangle.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug

subset.cpp 123pattern.c pascal triangle.c

```
1 #include <stdio.h>
2 int main() {
3     int rows, coef = 1, space, i, j;
4     printf("Enter the number of rows: ");
5     scanf("%d", &rows);
6     for (i = 0; i < rows; i++) {
7         for (space = 1; space <= rows - i; space++)
8             printf(" ");
9         for (j = 0; j < i; j++) {
10            if (j == 0 || i == 0)
11                coef = 1;
12            else
13                coef = coef * (i - j + 1) / j;
14            printf("%4d", coef);
15        }
16        printf("\n");
17    }
18    return 0;
19 }
```

C:\Users\VIKTUS\Desktop\DA x +

Enter the number of rows: 5

1
1 2 1
1 3 3 1
1 4 6 4 1

Process exited after 1.31 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

- Warnings: 0
- Output Filename: C:\Users\VIKTUS\Desktop\DA\pascal triangle.exe
- Output Size: 120.7794375 Kib
- Compilation Time: 0.17s

Shorten compiler paths

Line: 20 Cok: 1 Set: 0 Lines: 20 Length: 479 Insert Done parsing in 0 seconds

35°C Partly sunny

Search

14/22 03-08-2023

7. Write a program to find the optimal cost by using appropriate algorithm

PROGRAM

```

#include <stdio.h>
#include <limits.h>
#define MAX_KEYS 10
int optimalCostBST(int keys[], int freq[], int n) {
    int cost[n][n];
    for (int i = 0; i < n; i++) {
        cost[i][i] = freq[i];
    }
    for (int chain_len = 2; chain_len <= n; chain_len++) {
        for (int start = 0; start < n - chain_len + 1; start++) {
            int end = start + chain_len - 1;
            cost[start][end] = INT_MAX;
            for (int root = start; root <= end; root++) {
                int left_cost = (root > start) ? cost[start][root - 1] : 0;
                int right_cost = (root < end) ? cost[root + 1][end] : 0;
                int current_cost = left_cost + right_cost + freq[root];
                if (current_cost < cost[start][end]) {
                    cost[start][end] = current_cost;
                }
            }
        }
    }
}

```

```

        return cost[0][n - 1];
    }
int main() {
    int keys[MAX_KEYS], freq[MAX_KEYS];
    int n;
    printf("Enter the number of keys (maximum %d): ", MAX_KEYS);
    scanf("%d", &n);
    printf("Enter the keys in sorted order:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &keys[i]);
    }
    printf("Enter the frequencies of the keys:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &freq[i]);
    }
    int optimal_cost = optimalCostBST(keys, freq, n);
    printf("Optimal cost of the Binary Search Tree is: %d\n", optimal_cost);
    return 0;
}

```

OUTPUT

The screenshot shows a Windows desktop environment. On the left, there is a Dev-C++ IDE window titled 'C:\Users\VICTUS\Desktop\DA\Optimal costb of BST.cpp - [Executing] - Dev-C++ 5.11'. The code editor displays the C++ program for calculating the optimal cost of a binary search tree. On the right, there is a terminal window titled 'C:\Users\VICTUS\Desktop\DA'. The terminal output is as follows:

```

Enter the number of keys (maximum 10): 5
Enter the Keys in sorted order:
4 6 8 9 20
Enter the frequencies of the keys:
22 44 66 88 12
Optimal cost of the Binary Search Tree is: 232

```

Below the terminal window, the taskbar shows various icons including a weather icon for '35°C Haze', a search bar, and system status icons.

8. Write a program to find the sum of digits.

PROGRAM

```
#include<stdio.h>
```

```

int main()
{
int n,sum=0,m;
printf("Enter a number:");
scanf("%d",&n);
while(n>0)
{
m=n%10;
sum=sum+m;
n=n/10;
}
printf("Sum is=%d",sum);
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The code editor displays the following C program:

```

1 #include<stdio.h>
2 int main()
3 {
4     int n,sum=0,m;
5     printf("Enter a number:");
6     scanf("%d",&n);
7     while(n>0)
8     {
9         m=n%10;
10        sum=sum+m;
11        n=n/10;
12    }
13    printf("Sum is=%d",sum);
14    return 0;
15 }

```

The output window shows the execution results:

```

Enter a number:56
Sum is=11
Process exited after 3.254 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom indicates the system temperature is 37°C and the date is 03-08-2023.

9. Write a program to print minimum and maximum value sequence for all the numbers in a list.

PROGRAM

```

#include <stdio.h>
#include <conio.h>
int main()
{
    int a[1000],i,n,min,max;
    printf("Enter size of the array : ");

```

```

scanf("%d",&n);
printf("Enter elements in array : ");
for(i=0; i<n; i++)
{
    scanf("%d",&a[i]);
}
min=max=a[0];
for(i=1; i<n; i++)
{
    if(min>a[i])
        min=a[i];
    if(max<a[i])
        max=a[i];
}
printf("minimum of array is : %d",min);
printf("\nmaximum of array is : %d",max);
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. The project navigation bar at the top lists several files: subset.cpp, 123 pattern.c, pascaltriangle.c, number pattern.c, reverse a number.c, perfect number.cpp, floyds algorithm.cpp, sum of the digits.c, TSP.c, Optimal cost of BST.cpp, and max and min value sequence.c. The main code area contains the provided C program. The terminal window below shows the execution process:

```

C:\Users\VICTUS\Desktop\DA\max and min value sequence.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals) + v
Project Classes Debug subset.cpp 123 pattern.c pascaltriangle.c number pattern.c reverse a number.c perfect number.cpp floyds algorithm.cpp sum of the digits.c TSP.c Optimal cost of BST.cpp max and min value sequence.c
1 #include <stdio.h>
2 #include <conio.h>
3 int main()
4 {
5     int a[1000],i,n,min,max;
6     printf("Enter size of the array : ");
7     scanf("%d",&n);
8     printf("Enter elements in array : ");
9     for(i=0; i<n; i++)
10    {
11        scanf("%d",&a[i]);
12    }
13    min=max=a[0];
14    for(i=1; i<n; i++)
15    {
16        if(min>a[i])
17            min=a[i];
18        if(max<a[i])
19            max=a[i];
20    }
21    printf("minimum of array is : %d",min);
22    printf("\nmaximum of array is : %d",max);
23    return 0;
24 }

Enter size of the array : 5
Enter elements in array : 1
2
3
4
5
minimum of array is : 1
maximum of array is : 5
Process exited after 6.815 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom of the IDE provides system information like temperature (35°C), battery level, and date/time (03-08-2023).

10. Write a program to perform n Queens Problem using backtracking.

PROGRAM

```

#include<stdio.h>
#include<math.h>
int board[20],count;

```

```

int main()
{
int n,i,j;
void queen(int row,int n);
printf(" - N Queens Problem Using Backtracking -");
printf("\n\nEnter number of Queens:");
scanf("%d",&n);
queen(1,n);
return 0;
}
void print(int n)
{
int i,j;
printf("\n\nSolution %d:\n\n",++count);
for(i=1;i<=n;++i)
printf("\t%d",i);
for(i=1;i<=n;++i)
{
printf("\n\n%d",i);
for(j=1;j<=n;++j)
{
if(board[i]==j)
printf("\tQ");
else
printf("\t-");
}
}
}
int place(int row,int column)
{
int i;
for(i=1;i<=row-1;++i)
{
if(board[i]==column)
return 0;
else
if(abs(board[i]-column)==abs(i-row))
return 0;
}
return 1;
}
void queen(int row,int n)
{
int column;

```

```

for(column=1;column<=n;++column)
{
    if(place(row,column))
    {
        board[row]=column;
        if(row==n)
            print(n);
        else
            queen(row+1,n);
    }
}
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface with the code for the N-Queens problem. The code is a C program named 'n-queen.c' that uses backtracking to find solutions for placing queens on an n×n chessboard. It includes header files for stdio.h and math.h, defines a board of size 20x20, and initializes it with zeros. The main function reads the number of queens from the user, initializes variables, and calls the queen function. The queen function places queens in columns, checks for conflicts, and prints valid solutions. Two solutions are found for n=4:

```

Enter number of Queens:4

Solution 1:
      1   2   3   4
1   -   Q   -   -
2   -   -   -   Q
3   Q   -   -   -
4   -   -   Q   -
Solution 2:
      1   2   3   4
1   -   -   Q   -
2   Q   -   -   -
3   -   -   -   Q
4   -   Q   -   -

```

Process exited after 3.363 seconds with return value 0
Press any key to continue . . .

The status bar at the bottom shows the date and time as 03-08-2023 09:15.

CSA0683-DESIGN AND ANALYSIS OF ALGORITHMS FOR SEARCHING ALGORITHMS

B.Nikhil(192111054)

DAY-4

1. Write C program to insert a number in a list.

PROGRAM

```
#include<stdio.h>
int main()
{
    int student[40],pos,i,size,value;
    printf("enter no of elements in array of students:");
    scanf("%d",&size);
    printf("enter %d elements are:\n",size);
    for(i=0;i<size;i++)
        scanf("%d",&student[i]);
    printf("enter the position where you want to insert the element:");
    scanf("%d",&pos);
    printf("enter the value into that position:");
    scanf("%d",&value);
    for(i=size-1;i>=pos-1;i--)
        student[i+1]=student[i];
    student[pos-1]= value;
    printf("final array after inserting the value is\n");
    for(i=0;i<=size;i++)
        printf(" %d ",student[i]);
    return 0;
}
```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program named 'Insert element in a list.c'. The code prompts the user to enter the number of elements, the elements themselves, the position where to insert a new element, and the value to insert. It then prints the final array. On the right, the terminal window shows the execution of the program. The user enters 4 elements (10, 20, 30, 40), chooses to insert at position 3, and enters the value 80. The final array is printed as 10 20 80 30 40.

```

1 #include <stdio.h>
2 int main()
3 {
4     int student[40], pos, i, size, value;
5     printf("enter no of elements in array of students:");
6     scanf("%d", &size);
7     printf("enter %d elements are:\n", size);
8     for(i=0;i<size;i++)
9         | scanf("%d", &student[i]);
10    printf("enter the position where you want to insert the element:");
11    scanf("%d", &pos);
12    printf("enter the value into that position:");
13    scanf("%d", &value);
14    for(i=size-1;i>=pos-1;i--)
15        | public int __cdecl printf (const char * __restrict__ _Format, ...)
16        | student[pos-1]= value;
17        | printf("final array after inserting the value is\n");
18        | for(i=0;i<size;i++)
19        |     | printf(" %d ",student[i]);
20        | return 0;
21 }

```

2. Write C program to perform a sum of subsets problem using backtracking.

PROGRAM

```

#include <stdio.h>
#include <stdbool.h>
void generateSubsets(int arr[], int n, bool subset[], int index, int targetSum, int currentSum) {
    if (index == n) {
        if (currentSum == targetSum) {
            printf("Subset with target sum %d: {}", targetSum);
            for (int i = 0; i < n; i++) {
                if (subset[i]) {
                    printf(" %d", arr[i]);
                }
            }
            printf(" }\n");
        }
        return;
    }
    subset[index] = true;
    generateSubsets(arr, n, subset, index + 1, targetSum, currentSum + arr[index]);
    subset[index] = false;
    generateSubsets(arr, n, subset, index + 1, targetSum, currentSum);
}
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements:\n");

```

```

for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}
int targetSum;
printf("Enter the target sum: ");
scanf("%d", &targetSum);
bool subset[n];
generateSubsets(arr, n, subset, 0, targetSum, 0);
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code for 'Sum os subsets.cpp' is displayed. It includes headers for stdio.h and stdbool.h, defines MAX_VERTICES as 20, and contains functions for printing solutions and checking if a vertex is safe (i.e., it has no adjacent vertices with the same color). The main function prompts the user for the number of elements and the target sum, then calls the generateSubsets function.

The right side of the screen shows the terminal window where the program is running. It asks for the number of elements (5) and the target sum (10). It then lists three valid subsets: {1 2 3 4}, {1 4 5}, and {2 3 5}. Finally, it prints a message indicating the process exited after 6.113 seconds.

3. Write C program to perform graph colouring problem using backtracking.

PROGRAM

```

#include <stdio.h>
#include <stdbool.h>
#define MAX_VERTICES 20
void printSolution(int colors[], int num_vertices);
bool isSafe(int vertex, int graph[][MAX_VERTICES], int colors[], int num_vertices, int color) {
    for (int i = 0; i < num_vertices; i++) {
        if (graph[vertex][i] && colors[i] == color) {
            return false;
        }
    }
    return true;
}

```

```

bool graphColoringUtil(int graph[][MAX_VERTICES], int num_vertices, int m, int colors[], int vertex) {
    if (vertex == num_vertices) {
        return true;
    }
    for (int color = 1; color <= m; color++) {
        if (isSafe(vertex, graph, colors, num_vertices, color)) {
            colors[vertex] = color;
            if (graphColoringUtil(graph, num_vertices, m, colors, vertex + 1)) {
                return true;
            }
            colors[vertex] = 0;
        }
    }
    return false;
}
bool graphColoring(int graph[][MAX_VERTICES], int num_vertices, int m) {
    int colors[MAX_VERTICES];
    for (int i = 0; i < num_vertices; i++) {
        colors[i] = 0;
    }
    if (!graphColoringUtil(graph, num_vertices, m, colors, 0)) {
        return false;
    }
    printf("Graph can be colored using %d colors.\n", m);
    printf("Coloring of vertices:\n");
    printSolution(colors, num_vertices);
    return true;
}
void printSolution(int colors[], int num_vertices) {
    for (int i = 0; i < num_vertices; i++) {
        printf("Vertex %d: Color %d\n", i, colors[i]);
    }
}
int main() {
    int num_vertices, m;
    printf("Enter the number of vertices (max %d): ", MAX_VERTICES);
    scanf("%d", &num_vertices);
    int graph[MAX_VERTICES][MAX_VERTICES];
    printf("Enter the adjacency matrix for the graph:\n");
    for (int i = 0; i < num_vertices; i++) {
        for (int j = 0; j < num_vertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    printf("Enter the number of colors: ");
    scanf("%d", &m);
}

```

```
if (m < 1) {
    printf("Number of colors should be at least 1.\n");
    return 1;
}
if (!graphColoring(graph, num_vertices, m)) {
    printf("Graph cannot be colored with the given constraints.\n");
}
return 0;
}
```

OUTPUT

C:\Users\VIKTUS\Desktop\DA\Graph Colouring.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals) Graph Colouring.cpp Untitled3

Project Classes Debug

```
1 #include <stdio.h>
2 #include <stdbool.h>
3 #define MAX_VERTICES 20
4 void printSolution(int colors[], int num_vertices);
5 bool isSafe(int vertex, int graph[][MAX_VERTICES], int colors[], int num_vertices,
6             int m, int color) {
7     for (int i = 0; i < num_vertices; i++) {
8         if (graph[vertex][i] && colors[i] == color) {
9             return false;
10        }
11    }
12    return true;
13 }
14 bool graphColoringUtil(int graph[][MAX_VERTICES], int num_vertices, int m, int colors[])
15 {
16     if (vertex == num_vertices) {
17         return true;
18     }
19     for (int color = 1; color <= m; color++) {
20         if (isSafe(vertex, graph, colors, num_vertices, color)) {
21             colors[vertex] = color;
22             if (graphColoringUtil(graph, num_vertices, m, colors, vertex + 1)) {
23                 return true;
24             }
25         }
26     }
27     return false;
28 }
```

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

Shorten compiler paths

Line: 68 Col: 2 Sel: 0 Lines: 68 Length: 2236 Insert Done parsing in 0.015 seconds

3% Haze

(C:\Users\VIKTUS\Desktop\DA\Graph Colouring.exe)

Search

C:\Users\VIKTUS\Desktop\DA

Enter the number of vertices (max 20): 5
Enter the adjacency matrix for the graph:
0 1 1 0 0
1 0 1 0 0
1 1 0 1 1
0 0 1 0 1
0 0 1 1 0
Enter the number of colors: 3
Graph can be colored using 3 colors.
Coloring of vertices:
Vertex 0: Color 1
Vertex 1: Color 2
Vertex 2: Color 3
Vertex 3: Color 1
Vertex 4: Color 2

Process exited after 15.36 seconds with return value 0
Press any key to continue . . . |

0905

4. Write C program to compute container loader Problem.

PROGRAM

```
#include <stdio.h>
#define MAX_CONTAINERS 100
#define MAX_ITEMS 100
void containerLoading(int containers[], int numContainers, int items[], int numItems) {
    int capacity[MAX_CONTAINERS];
    for (int i = 0; i < numContainers; i++) {
        capacity[i] = containers[i];
    }
    for (int i = 0; i < numItems; i++) {
        int item = items[i];
        int j;
        for (j = 0; j < numContainers; j++) {
```

```

        if (capacity[j] >= item) {
            printf("Item %d fits in Container %d\n", item, j + 1);
            capacity[j] -= item;
            break;
        }
    }
    if (j == numContainers) {
        printf("Item %d cannot be fit into any container. It exceeds the capacity.\n", item);
    }
}
int main() {
    int containers[MAX_CONTAINERS];
    int numContainers;
    printf("Enter the number of containers: ");
    scanf("%d", &numContainers);
    printf("Enter the capacities of %d containers:\n", numContainers);
    for (int i = 0; i < numContainers; i++) {
        scanf("%d", &containers[i]);
    }
    int items[MAX_ITEMS];
    int numItems;
    printf("Enter the number of items: ");
    scanf("%d", &numItems);
    printf("Enter the sizes of %d items:\n", numItems);
    for (int i = 0; i < numItems; i++) {
        scanf("%d", &items[i]);
    }
    printf("\nContainer Loading Result:\n");
    containerLoading(containers, numContainers, items, numItems);
    return 0;
}

```

OUTPUT

C:\Users\VIKTUS\Desktop\DA\Container loader program.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

Project: Classes Debug Hamiltonian circuit.cpp Graph Colouring.cpp Container loader program.cpp

File containerLoading(int containers[], int numContainers, int items[], int numItems); main(): int

```
4 void containerLoading(int containers[], int numContainers, int items[], int numItems) {
5     int capacity[MAX_CONTAINERS];
6     for (int i = 0; i < numContainers; i++) {
7         capacity[i] = containers[i];
8     }
9     for (int i = 0; i < numItems; i++) {
10        int item = items[i];
11        int j;
12        for (j = 0; j < numContainers; j++) {
13            if (capacity[j] >= item) {
14                printf("Item %d fits in Container %d\n", item, j + 1);
15                capacity[j] -= item;
16                break;
17            }
18        }
19        if (j == numContainers) {
20            printf("Item %d cannot be fit into any container. It exceeds the capacity.\n", item);
21        }
22    }
23 }
24 int main() {
25     int containers[MAX_CONTAINERS];
26     int numContainers;
27     printf("Enter the number of containers: ");
28     scanf("%d", &numContainers);
29     printf("Enter the capacities of %d containers:\n", numContainers);
30     for (int i = 0; i < numContainers; i++) {
31         scanf("%d", &containers[i]);
32     }
33     int items[MAX_ITEMS];
34     int numItems;
35     printf("Enter the number of items: ");
36     scanf("%d", &numItems);
37     printf("Enter the sizes of %d items:\n", numItems);
38     for (int i = 0; i < numItems; i++) {
39         scanf("%d", &items[i]);
40     }
41     printf("\nContainer Loading Result:\n");
42     containerLoading(containers, numContainers, items, numItems);
43     return 0;
44 }
```

C:\Users\VIKTUS\Desktop\DA x +

Enter the number of containers: 5
Enter the capacities of 5 containers:
10
20
30
40
50
Enter the number of items: 5
Enter the sizes of 5 items:
6
12
25
38
45

Container Loading Result:
Item 6 fits in Container 1
Item 12 fits in Container 2
Item 25 fits in Container 3
Item 38 fits in Container 4
Item 45 fits in Container 5

Process exited after 28.78 seconds with return value 0
Press any key to continue . . .

Compiler Resources Compile Log Debug Find Results

Line: 34 Cok: 18 Sel: 0 Lines: 44 Length: 1488 Insert Done parsing in 0 seconds

Breaking news ENG IN 14:14 04-08-2023

5. Write a program to generate the list of all factors for n value.

PROGRAM

```
#include <stdio.h>
int main()
{
    int i, num;
    printf("Enter any number to find its factor: ");
    scanf("%d", &num);
    printf("All factors of %d are: \n", num);
    for(i=1; i<=num; i++)
    {
        if(num % i == 0)
        {
            printf("%d, ", i);
        }
    }
    return 0;
}
```

OUTPUT

```

1 #include <stdio.h>
2 int main()
3 {
4     int i, num;
5     printf("Enter any number to find its factor: ");
6     scanf("%d", &num);
7     printf("All factors of %d are: \n", num);
8     for(i=1; i<=num; i++)
9     {
10         if(num % i == 0)
11         {
12             printf("%d, ", i);
13         }
14     }
15     return 0;
16 }

```

6. Write a program to perform Assignment problem using branch and bound

PROGRAM

```

#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
#define INF 9999
int numWorkers;
int numJobs;
int costMatrix[MAX_SIZE][MAX_SIZE];
bool assigned[MAX_SIZE];
int minCost = INF;
int finalAssignment[MAX_SIZE];
void printAssignment() {
    printf("Optimal Assignment:\n");
    for (int i = 0; i < numWorkers; i++) {
        printf("Worker %d -> Job %d\n", i + 1, finalAssignment[i] + 1);
    }
}
void updateMinCost(int assignment[MAX_SIZE]) {
    int totalCost = 0;
    for (int i = 0; i < numWorkers; i++) {
        totalCost += costMatrix[i][assignment[i]];
    }
}

if (totalCost < minCost) {

```

```

minCost = totalCost;
for (int i = 0; i < numWorkers; i++) {
    finalAssignment[i] = assignment[i];
}
}

void branchAndBound(int worker, int currentCost, int assignment[MAX_SIZE]) {
    if (worker == numWorkers) {
        updateMinCost(assignment);
        return;
    }
    for (int job = 0; job < numJobs; job++) {
        if (!assigned[job]) {
            int newCost = currentCost + costMatrix[worker][job];

            if (newCost < minCost) {
                assigned[job] = true;
                assignment[worker] = job;

                branchAndBound(worker + 1, newCost, assignment);

                assigned[job] = false;
            }
        }
    }
}

int main() {
    printf("Enter the number of workers: ");
    scanf("%d", &numWorkers);
    printf("Enter the number of jobs: ");
    scanf("%d", &numJobs);
    printf("Enter the cost matrix (%d x %d):\n", numWorkers, numJobs);
    for (int i = 0; i < numWorkers; i++) {
        for (int j = 0; j < numJobs; j++) {
            scanf("%d", &costMatrix[i][j]);
        }
    }
    int assignment[MAX_SIZE];
    for (int i = 0; i < numWorkers; i++) {
        assigned[i] = false;
    }
    branchAndBound(0, 0, assignment);
    printf("Minimum Cost: %d\n", minCost);
    printAssignment();
}

```

```

        return 0;
}

```

OUTPUT

```

C:\Users\VICTUS\Desktop\DA\Assignment problem using branch and bound.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug Hamiltonian circuit.cpp Graph Colouring.cpp Container loader program.cpp list the factors for n.c Assignment problem using branch and bound.cpp
branchAndBound(n
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX_SIZE 100
4 #define INF 9999
5 int numWorkers;
6 int numJobs;
7 int costMatrix[MAX_SIZE][MAX_SIZE];
8 bool assigned[MAX_SIZE];
9 int minCost = INF;
10 int finalAssignment[MAX_SIZE];
11 void printAssignment() {
12     printf("Optimal Assignment:\n");
13     for (int i = 0; i < numWorkers; i++) {
14         printf("Worker %d --> Job %d\n", i + 1, finalAssignment[i] + 1);
15     }
16 }
17 void updateMinCost(int assignment[MAX_SIZE]) {
18     int totalCost = 0;
19     for (int i = 0; i < numWorkers; i++) {
20         totalCost += costMatrix[i][assignment[i]];
21     }
22     if (totalCost < minCost) {
23         minCost = totalCost;
24         for (int i = 0; i < numWorkers; i++) {
25             finalAssignment[i] = assignment[i];
26         }
27     }
28 }
29 void branchAndBound(int worker, int currentCost, int assignment[MAX_SIZE]) {
30     if (worker == numWorkers) {
31         updateMinCost(assignment);
32         return;
33     }
34     for (int job = 0; job < numJobs; job++) {
35         if (!assigned[job]) {
36             int newCost = currentCost + costMatrix[worker][job];
37             if (newCost < minCost) {
38                 assigned[job] = true;
39                 assignment[worker] = job;
40             }
41         }
42     }
43 }
44
45 int main() {
46     updateMinCost(assigned);
47     for (int i = 0; i < numWorkers; i++) {
48         assigned[i] = false;
49     }
50     branchAndBound(0, 0, finalAssignment);
51     printAssignment();
52     printf("Minimum Cost: %d\n", minCost);
53 }

```

Enter the number of workers: 4
Enter the number of jobs: 4
Enter the cost matrix (4 x 4):
2 5 8 2
5 6 8 7
9 5 3 6
0 4 8 7
Minimum Cost: 11
Optimal Assignment:
Worker 1 --> Job 4
Worker 2 --> Job 2
Worker 3 --> Job 3
Worker 4 --> Job 1

Process exited after 77.96 seconds with return value 0
Press any key to continue . . .

7. Write a program to perform linear search.

PROGRAM

```

#include <stdio.h>
int main()
{
    int array[100], search, c, n;
    printf("Enter number of elements in array\n");
    scanf("%d", &n);
    printf("Enter %d integer(s)\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    printf("Enter a number to search\n");
    scanf("%d", &search);
    for (c = 0; c < n; c++)
    {
        if (array[c] == search)
        {
            printf("%d is present at location %d.\n", search, c+1);
            break;
        }
    }
}

```

```

    }
if (c == n)
    printf("%d isn't present in the array.\n", search);
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C program for linear search. On the right, a terminal window shows the execution of the program. The user enters 5 integers: 18, 20, 30, 40, 80. Then, they enter a number to search for: 60. The program outputs that 60 is not present in the array. The terminal also shows the process exit status and a prompt to press any key to continue.

```

C:\Users\VICTUS\Desktop\DA> Enter number of elements in array
5
Enter 5 integer(s)
18 20 30 40 80
Enter a number to search
60
60 isn't present in the array.

Process exited after 10.88 seconds with return value 0
Press any key to continue . . .

```

8. Write a program to find out Hamiltonian circuit Using backtracking method

PROGRAM

```

#include <stdio.h>
#include <stdbool.h>
#define MAX_VERTICES 100
int graph[MAX_VERTICES][MAX_VERTICES];
int numVertices;
int hamiltonianPath[MAX_VERTICES];
bool isSafe(int v, int pos, int path[MAX_VERTICES]) {
    if (graph[path[pos - 1]][v] == 0) {
        return false;
    }
    for (int i = 0; i < pos; i++) {
        if (path[i] == v) {
            return false;
        }
    }
}

```

```

        return true;
    }
bool hamiltonianUtil(int path[MAX_VERTICES], int pos) {
    if (pos == numVertices) {
        if (graph[path[pos - 1]][path[0]] == 1) {
            return true;
        }
        return false;
    }
    for (int v = 1; v < numVertices; v++) {
        if (isSafe(v, pos, path)) {
            path[pos] = v;
            if (hamiltonianUtil(path, pos + 1)) {
                return true;
            }
            path[pos] = -1;
        }
    }
    return false;
}
bool hamiltonianCircuit() {
    int path[MAX_VERTICES];
    for (int i = 0; i < numVertices; i++) {
        path[i] = -1;
    }
    path[0] = 0;
    if (hamiltonianUtil(path, 1)) {
        for (int i = 0; i < numVertices; i++) {
            hamiltonianPath[i] = path[i];
        }
        hamiltonianPath[numVertices] = path[0];
        return true;
    }
    return false;
}
int main() {
    printf("Enter the number of vertices in the graph: ");
    scanf("%d", &numVertices);
    printf("Enter the adjacency matrix of the graph (1 if there's an edge, 0 otherwise):\n");
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
}

```

```

if (hamiltonianCircuit()) {
    printf("Hamiltonian Circuit Found:\n");
    for (int i = 0; i <= numVertices; i++) {
        printf("%d ", hamiltonianPath[i]);
    }
    printf("\n");
} else {
    printf("Hamiltonian Circuit Not Found.\n");
}
return 0;
}

```

OUTPUT

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ file named `Hamiltonian circuit.cpp`. The code implements a backtracking algorithm to find a Hamiltonian circuit in a graph. On the right, a terminal window titled `C:\Users\...\\DA` shows the execution of the program. It prompts for the number of vertices (5) and the adjacency matrix. The matrix is represented as:

```

0 1 1 1 0
1 0 1 0 1
1 1 0 1 1
1 0 1 0 1
0 1 1 1 0

```

The program outputs "Hamiltonian Circuit Found:" followed by the path `0 1 2 4 3 0`. The terminal also shows the process exit time and a prompt to press any key.