

DAA PRACTICAL PROGRAMS**1. Fibonacci number using recursion**

```
#include<stdio.h>

int fibonacci(int n)
{
    if(n==0)
    {
        return 0;
    }
    else if(n==1)
    {
        return 1;
    }
    else
    {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}

int main()
{
    int i,n;
    printf("Enter the n value:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("%d ",fibonacci(i));
    }
}
```

```
}
```

```
}
```

The screenshot shows the Dev-C++ IDE interface. The project file 'fibonacci recursion.cpp' is open in the editor. The code implements a recursive function to calculate Fibonacci numbers. The main() function prompts the user for an input value, prints the sequence up to that value, and then exits.

```
#include<stdio.h>
int fibonacci(int n)
{
    if(n==0)
    {
        return 0;
    }
    else if(n==1)
    {
        return 1;
    }
    else
    {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}
int main()
{
    int i,n;
    printf("Enter the n value:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        public int __cdecl printf (const char * __restrict
                                    ,printf("%d ",fibonacci(i));
    }
}
```

The terminal window shows the following output:

```
Enter the n value:10
0 1 1 2 3 5 8 13 21 34
-----
Process exited after 1.773 seconds with return value 0
Press any key to continue . . .
```

The status bar at the bottom provides compilation information:

```
Line: 24 Col: 20 Sel: 0 Lines: 26 Length: 309 Insert Done parsing in 0.016 seconds
```

2. Armstrong number

```
#include<stdio.h>

int armstrong(int n)
{
    int sum=0,r,temp=n;
    while(n>0)
    {
        r=n%10;
        sum=sum+(r*r*r);
    }
}
```

```
n=n/10;  
}  
if(sum==temp)  
{  
    return 1;  
}  
else  
{  
    return -1;  
}  
}  
  
int main()  
{  
    int n;  
    printf("Enter the number:");  
    scanf("%d",&n);  
    int result=armstrong(n);  
    if(result==1)  
    {  
        printf("%d is armstrong number",n);  
    }  
    else  
{  
        printf("%d is not armstrong number",n);  
    }  
    return 0;
```

{

The screenshot shows the Dev-C++ IDE interface. The left pane displays the code for 'armstrong number.cpp'. The right pane shows the terminal window output. The code implements a function to check if a number is Armstrong, and the terminal shows the program's execution and output for the number 153.

```
#include<stdio.h>
int armstrong(int n)
{
    int sum=0,r,temp=n;
    while(n>0)
    {
        r=n%10;
        sum=sum+(r*r*r);
        n=n/10;
    }
    if(sum==temp)
    {
        return 1;
    }
    else
    {
        return -1;
    }
}
int main()
{
    int n;
    printf("Enter the number:");
    scanf("%d",&n);
}
```

Output:

```
Enter the number:153
153 is armstrong number
-----
Process exited after 3.684 seconds with return value 0
Press any key to continue . . .
```

3. GCD

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

```
int main()
```

```
{
```

```
    int i,a,b,gcd;
```

```

printf("Enter the numbers:");

scanf("%d %d",&a,&b);

for(i=1;i<=a && i<=b;++i)

{

if(a%i==0&&b%i==0)

    {

gcd=i;

    }

}

printf("%d",gcd);

}

```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ file named GCD.cpp with the following content:

```

1 #include<stdio.h>
2 int main()
3 {
4     int i,a,b,gcd;
5     printf("Enter the numbers:");
6     scanf("%d %d",&a,&b);
7     for(i=1;i<=a && i<=b;++i)
8     {
9         if(a%i==0&&b%i==0)
10        {
11            gcd=i;
12        }
13    }
14    printf("%d",gcd);
15 }

```

To the right of the code editor is a terminal window titled "C:\Users\mandl\C++\GCD.exe" showing the output of the program:

```

Enter the numbers:72
36
36
-----
Process exited after 9.215 seconds with return value 0
Press any key to continue . .

```

Below the terminal window is a "Compiler" tab showing compilation results:

- Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\C++\GCD.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.27s

The status bar at the bottom shows the current line (Line: 7), column (Col: 28), and total lines (Lines: 15). It also indicates "Done parsing in 0.015 seconds".

4. Largest number in an array

```

#include<stdio.h>
int main()
{

```

```

int a[50],i,n,max;
printf("Enter the size of array:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
    scanf("%d",&a[i]);
}
max=a[0];
for(i=1;i<n;i++)
{
    if(max<a[i])
    {
        max=a[i];
    }
}
printf("%d is largest number",max);
}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays the code for finding the largest number in an array. The code uses a for loop to read array elements and an inner if loop to update the maximum value. The output window shows the user input (size 6, elements 15, 18, 21, 5, 2) and the program's output (21 is largest number). The status bar at the bottom provides compilation and execution details.

Code:

```

1 #include<stdio.h>
2 int main()
3 {
4     int a[50],i,n,max;
5     printf("Enter the size of array:");
6     scanf("%d",&n);
7     printf("Enter the elements:");
8     for(i=0;i<n;i++)
9     {
10         scanf("%d",&a[i]);
11     }
12     max=a[0];
13     for(i=1;i<n;i++)
14     {
15         if(max<a[i])
16         {
17             max=a[i];
18         }
19     }
20     printf("%d is largest number",max);
21 }

```

Output:

```

Enter the size of array:6
Enter the elements:15
18
21
5
2
21 is largest number

```

Compiler Log:

```

Compilation results...
=====
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\C++\largest numbe
- Output Size: 128.62109375 Kib
- Compilation Time: 0.34s

```

Status Bar:

Line: 3 Col: 2 Sel: 0 Lines: 21 Length: 313 Insert Done parsing in 0.015 seconds WhatsApp

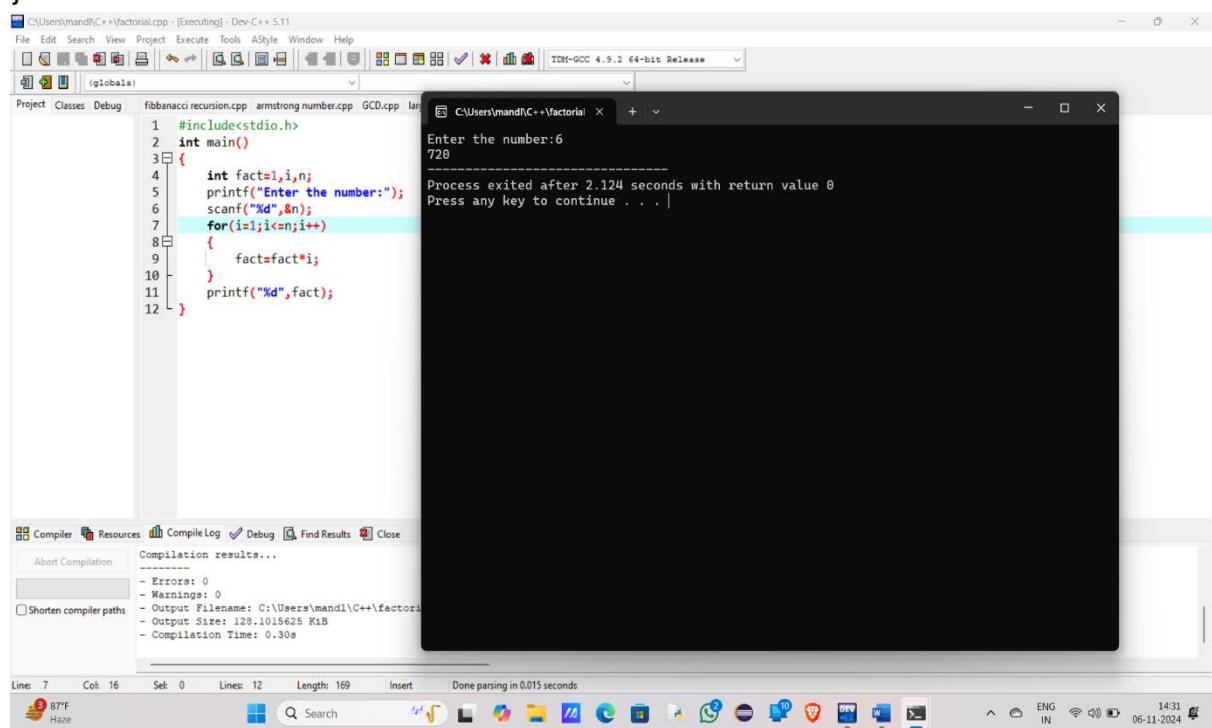
Very humid Now

Search

ENG IN 1425 06-11-2024

5. Factorial of a number

```
#include<stdio.h>
int main()
{
    int fact=1,i,n;
    printf("Enter the number:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        fact=fact*i;
    }
    printf("%d",fact);
}
```



6. Prime Number:

```
#include<stdio.h>
int main()
{
    int i,n,count=0;
    printf("Enter the number:");
    scanf("%d",&n);
    for(i=1;i<n;i++)
    {
```

```

if(n%i==0)
{
    count=count+1;
}
else
{
    continue;
}
}
if(count==1)
{
    printf("%d is prime number",n);
}
else
{
    printf("not a prime number");
}
}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C++ source code for a prime number checker. The code uses a for loop to iterate from 2 to n-1, checking if n is divisible by i. If count reaches 1, it's a prime number; otherwise, it's not. The output window shows the program's execution: it prompts for a number (7), outputs "7 is prime number", and then exits. The status bar at the bottom right shows the date and time as 06-11-2024 14:39.

7. To Perform Selection Section:

```
#include<stdio.h>

int main()
{
    int a[50],i,j,n,min,temp,k;
    printf("Enter the array size:");
    scanf("%d",&n);
    printf("Enter the array elements:");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++)
    {
        k=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[k])
            {
                k=j;
            }
        }
        temp=a[i];
        a[i]=a[k];
        a[k]=temp;
    }
    for(i=0;i<n;i++)

```

```
{
    printf("%d ",a[i]);
}

return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ file named 'selection sort.cpp' containing a selection sort algorithm. The terminal window shows the execution of the program, prompting for array size and elements, and then displaying the sorted array. The compiler log shows a successful compilation with no errors or warnings.

```
#include<stdio.h>
int main()
{
    int a[50],i,j,n,min,temp,k;
    printf("Enter the array size:");
    scanf("%d",&n);
    printf("Enter the array elements:");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++)
    {
        k=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[k])
            {
                k=j;
            }
        }
        temp=a[i];
        a[i]=a[k];
        a[k]=temp;
    }
}
```

Enter the array size:
Enter the array elements:15
2
8
1
21
21
4
1 2 4 8 15 21

Process exited after 20.68 seconds with return value 0
Press any key to continue . . .

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mand1\C++\selection sort.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.23s

8. Bubble sort:

```
#include<stdio.h>

int main()

{
    int a[50],n,i,j,temp;
    printf("Enter the array size:");
    scanf("%d",&n);
    printf("Enter the array elements:");
```

```
for(i=0;i<n;i++)
{
    scanf("%d",&a[i]);
}

for(i=0;i<n;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(a[i]>a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
}

for(i=0;i<n;i++)
{
    printf("%d ",a[i]);
}
}
```

```

C:\Users\mandl\Documents\Visual Studio 2022\Projects\bubble sort\bubble sort.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\mandl\Documents\Visual Studio 2022\Projects\bubble sort\bubble sort.cpp (globals)
Project Classes Debug selection sort.cpp bubble sort.cpp
1 #include<stdio.h>
2 int main()
3 {
4     int a[50],n,i,j,temp;
5     printf("Enter the array size:");
6     scanf("%d",&n);
7     printf("Enter the array elements:");
8     for(i=0;i<n;i++)
9     {
10         scanf("%d",&a[i]);
11     }
12     for(i=0;i<n;i++)
13     {
14         for(j=i+1;j<n;j++)
15         {
16             if(a[i]>a[j])
17             {
18                 temp=a[i];
19                 a[i]=a[j];
20                 a[j]=temp;
21             }
22         }
23     }
24     for(i=0;i<n;i++)
25 }

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation Compilation results...
=====
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\Documents\Visual Studio 2022\Projects\bubble sort\bubble sort.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.25s

Line: 28 Col: 2 Sel: 0 Lines: 28 Length: 382 Insert Done parsing in 0 seconds
29°C Mostly cloudy 13:17 ENG IN 07-11-2024

```

9. Matrix Multiplication:

```

#include<stdio.h>
int main()
{
    int a[50][50],b[50][50],c[50][50],i,j,k,n,m;
    printf("Enter the row size:");
    scanf("%d",&n);
    printf("Enter the column size:");
    scanf("%d",&m);
    printf("Enter the matrix a elements:");
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
}

```

```
printf("Enter the matrix b elements:");
for(i=0;i<n;i++)
{
    for(j=0;j<m;j++)
    {
        scanf("%d",&b[i][j]);
    }
}
for(i=0;i<n;i++)
{
    for(j=0;j<m;j++)
    {
        c[i][j]=0;
        for(k=0;k<n;k++)
        {
            c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
return 0;
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ source code file named 'matrix multiplication.cpp'. The code implements matrix multiplication between two 2x2 matrices. The user is prompted to enter the row and column sizes, followed by the elements of both matrices. The output window shows the entered values and the resulting matrix product.

```

1 #include<stdio.h>
2 int main()
3 {
4     int a[50][50],b[50][50],c[50][50],i,j;
5     printf("Enter the row size:");
6     scanf("%d",&n);
7     printf("Enter the column size:");
8     scanf("%d",&m);
9     printf("Enter the matrix a elements:");
10    for(i=0;i<n;i++)
11    {
12        for(j=0;j<m;j++)
13        {
14            scanf("%d",&a[i][j]);
15        }
16    }
17    printf("Enter the matrix b elements:");
18    for(i=0;i<n;i++)
19    {
20        for(j=0;j<m;j++)
21        {
22            scanf("%d",&b[i][j]);
23        }
24    }

```

Output window content:

```

Enter the row size:2
Enter the column size:2
Enter the matrix a elements:1 2
3
4
Enter the matrix b elements:5 6
7 8
19 22
43 50
-----
Process exited after 38.1 seconds with return value 0
Press any key to continue . . .

```

Compiler Log:

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\C++\matrix multiplication.cpp
- Output Size: 129.798828125 Kib
- Compilation Time: 0.24s

```

10.String is a palindrome:

```

#include<stdio.h>
#include<string.h>
int main()
{
    int i,l,j=0;
    char str[50],reverse[50];
    printf("Enter the string:");
    scanf("%s",str);
    l=strlen(str);
    for(i=l-1;i>=0;i--)
    {
        reverse[j]=str[i];
        j++;
    }
    reverse[j]='\0';
    if(strcmp(str,reverse)==0)
    {

```

```

        printf("%s is a palindrome",str);
    }
else
{
    printf("not a palindrome");
}
}

```

C:\Users\mandi\Documents\reverse a string.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

Project Classes Debug selection sort.cpp bubble sort.cpp matrix multiplication.cpp reverse a string.c

C:\Users\mandi\Documents\reverse a string.c

```

1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     int i,l,j=0;
6     char str[50],reverse[50];
7     printf("Enter the string:");
8     scanf("%s",str);
9     l=strlen(str);
10    for(i=l-1;i>=0;i--)
11    {
12        reverse[j]=str[i];
13        j++;
14    }
15    reverse[j]='\0';
16    if(strcmp(str,reverse)==0)
17    {
18        printf("%s is a palindrome",str);
19    }
20    else
21    {
22        printf("not a palindrome");
23    }
24 }

```

Enter the string:madam
madam is a palindrome

Process exited after 13.3 seconds with return value 0
Press any key to continue . . . |

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\Documents\reverse a string.
- Output Size: 128.79296875 KiB
- Compilation Time: 0.24s

Line: 15 Col: 17 Sel: 0 Lines: 24 Length: 365 Insert Done parsing in 0 seconds

20°C T-storms

Search

13:53 07-11-2024 ENG IN

11. Copy String:

```

#include<stdio.h>
#include<string.h>
int main()
{
    int i,l;
    char source[50],destination[50];
    printf("Enter the string:");
    scanf("%s",source);
    l=strlen(source);

```

```

for(i=0;i<l;i++)
{
    destination[i]=source[i];
}
destination[i]='\0';
printf("%s is the source ",source);
printf("%s is the destination",destination);
return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The main window displays the C code for copying a string. The code includes a for loop to copy characters from the source array to the destination array, followed by a null terminator. It also prints the source and destination strings. The output window shows the input 'dinesh' and the output 'dinesh is the source dinesh is the destination'. The status bar at the bottom indicates the system date and time.

```

C:\Users\mandl\copy string.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASStyle Window Help
C:\Users\mandl\copy string.cpp - [globals]
Project Classes Debug selection sort.cpp bubble sort.cpp matrix multiplication.cpp reverse a string.cen... copy string.cpp
1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     int i,l;
6     char source[50],destination[50];
7     printf("Enter the string:");
8     scanf("%s",source);
9     l=strlen(source);
10    for(i=0;i<l;i++)
11    {
12        destination[i]=source[i];
13    }
14    destination[i]='\0';
15    printf("%s is the source ",source);
16    public int __cdecl printf (const char * _rest
17    return 0;
18 }

C:\Users\mandl\copy str x + v
Enter the string:dinesh
dinesh is the source dinesh is the destination
Process exited after 2.823 seconds with return value 0
Press any key to continue . . .

```

Compiler Resources Compile Log Debug Find Results Close

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\copy string.exe
- Output Size: 128.1015625 KB
- Compilation Time: 0.25s

Line: 15 Col: 31 Sel: 0 Lines: 18 Length: 353 Insert Done parsing in 0 seconds

1 Rain to stop 256 pm 14:07 ENG IN 07-11-2024

12. Binary Search:

```

#include<stdio.h>
int main()
{
    int a[50],i,j,n,mid,end,start,temp,t,c=0;
    printf("Enter the size of array:\n");
    scanf("%d",&n);
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++)
    {

```

```
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(a[i]>a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
start=0;
end=n-1;
printf("Enter the element to find:\n");
scanf("%d",&t);
while(start<=end)
{
    mid=start+(end-start)/2;
    if(a[mid]==t)
    {
        c=1;
        printf("%d is found at %d index",t,mid);
        break;
    }
    else if(a[mid]<t)
    {
        start=mid+1;
    }
    else
    {
        end=mid-1;
    }
}
if(c==0)
```

```

{
    printf("not found");
}
return 0;
}

```

```

C:\Users\mandl\bin\binary search.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASyle Window Help
C:\Users\mandl\bin\binary s x + v
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug binary search.cpp
1 #include<stdio.h>
2 int main()
3 {
4     int a[50],i,j,n,mid;
5     printf("Enter the size:");
6     scanf("%d",&n);
7     printf("Enter the array elements:");
8     for(i=0;i<n;i++)
9     {
10        scanf("%d",&a[i]);
11    }
12    for(i=0;i<n;i++)
13    {
14        for(j=i+1;j<n;j++)
15        {
16            if(a[i]>a[j])
17            {
18                temp=a[i];
19                a[i]=a[j];
20                a[j]=temp;
21            }
22        }
23    }
24    start=0;
Enter the size of array:
9
Enter the array elements:
15
5
-3
16
30
10
23
12
Enter the element to find:
16
16 is found at 5 index
Process exited after 37.6 seconds with return value 0
Press any key to continue . . .

```

Compiler Resources Compile Log Debug Find Results

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\bin\binary search
- Output Size: 129.2705078125 KB
- Compilation Time: 0.30s

Line: 43 Col: 19 Sel: 0 Lines: 51 Length: 732 Insert Done parsing in 0 seconds

26°C Partly cloudy

Search

22:36 07-11-2024 ENG IN

13. Reverse a String:

```

#include<string.h>
int main()
{
    int i,l,j=0;
    char str[50],reverse[50];
    printf("Enter the string:");
    scanf("%s",str);
    l=strlen(str);
    for(i=l-1;i>=0;i--)
    {

```

```

        reverse[j]=str[i];
        j++;
    }
    reverse[j]='\0';
    printf("%s\n",reverse);
}

```

C:\Users\mandi\Documents\reverse_string.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools ASStyle Window Help

Project Classes Debug selection sort.cpp bubble sort.cpp matrix multiplication.cpp reverse_string.cpp

Enter the string:saveetha
ahseevas

Process exited after 5.928 seconds with return value 0
Press any key to continue . . .

#include<stdio.h>
#include<string.h>
int main()
{
 int i,l;
 char str[50];
 printf("Enter the string:");
 scanf("%s",str);
 l=strlen(str);
 for(i=l-1;i>=0;i--)
 {
 printf("%c",str[i]);
 }
 public int __cdecl printf (const char *

Compiler Resources Compile Log Debug Find Results Close

Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\Documents\reverse_string.exe
- Output Size: 128.7734375 Kib
- Compilation Time: 0.27s

Line: 12 Col: 19 Sel: 0 Lines: 14 Length: 205 Insert Done parsing in 0 seconds

30°C Light rain

Search

14:19 07-11-2024 ENG IN

14.Length of String:

```

#include<stdio.h>
int main()
{
    int i=0,l=0;
    char str[50];
    printf("Enter the string:");
    scanf("%s",str);
    while(str[i]!='\0')
    {

```

```

    i++;
    i++;
}
printf("%d",l);
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a file named 'length of string.cpp' with the following content:

```

1 #include<stdio.h>
2 int main()
3 {
4     int i=0,l=0;
5     char str[50];
6     printf("Enter the st");
7     scanf("%s",str);
8     while(str[i]!='\0')
9     {
10         l++;
11         i++;
12     }
13     printf("%d",l);
14 }

```

The output window shows the following interaction:

```

C:\Users\mandi\length of string.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug length of string.cpp
Enter the string:saveetha
8
Process exited after 12.65 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom shows the compilation results:

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\length of string.exe
- Output Size: 128.1220703125 KiB
- Compilation Time: 0.27s

```

15. Strassen's Matrix:

```

#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;
```

```
}
```

The screenshot shows the Dev-C++ IDE interface. The code in the editor is for multiplying two 2x2 matrices using Strassen's algorithm. The output window shows the user inputting matrices, the program outputting them, and the final result.

```

C:\Users\mandl\C++\strassen\matrix.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASyle Window Help
(globals) Project Classes Debug Untitled1 factorial.cpp prime number.cpp selection sort.c
C:\Users\mandl\C++\strassen x + v
1 #include<stdio.h>
2 int main()
3 {
4     int a[2][2], b[2][2], c[2][2];
5     int m1, m2, m3, m4, m5, m6, m7;
6
7     printf("Enter the 4 elements of first matrix: ");
8     for(i = 0; i < 2; i++)
9     {
10        for(j = 0; j < 2; j++)
11        {
12            scanf("%d", &a[i][j]);
13        }
14    }
15
16    printf("Enter the 4 elements of second matrix: ");
17    for(i = 0; i < 2; i++)
18    {
19        for(j = 0; j < 2; j++)
20        {
21            scanf("%d", &b[i][j]);
22        }
23    }
24
25    printf("\nThe first matrix is");
26    for(i = 0; i < 2; i++)
27    {
28        for(j = 0; j < 2; j++)
29        {
30            printf("%d\t", a[i][j]);
31        }
32    }
33    printf("\n\nThe second matrix is");
34    for(i = 0; i < 2; i++)
35    {
36        for(j = 0; j < 2; j++)
37        {
38            printf("%d\t", b[i][j]);
39        }
40    }
41
42    printf("\nMultiplication using Strassen's algorithm");
43
44    m1 = a[0][0]*b[0][0] + a[0][1]*b[1][0];
45    m2 = a[0][0]*b[0][1] + a[0][1]*b[1][1];
46    m3 = a[0][1]*b[0][0] + a[1][1]*b[1][0];
47    m4 = a[1][1]*b[0][1] + a[1][0]*b[1][1];
48    m5 = a[0][0]*b[0][0] + a[1][0]*b[1][0];
49    m6 = a[0][0]*b[0][1] + a[1][0]*b[1][1];
50    m7 = a[0][1]*b[0][0] + a[1][1]*b[1][0];
51
52    c[0][0] = m1 + m4 - m5 + m7;
53    c[0][1] = m2 + m4;
54    c[1][0] = m3 + m5;
55    c[1][1] = m1 + m2 - m3 + m6;
56
57    printf("\n\nThe result matrix is");
58    for(i = 0; i < 2; i++)
59    {
60        for(j = 0; j < 2; j++)
61        {
62            printf("%d\t", c[i][j]);
63        }
64    }
65
66    return 0;
}

```

Output window content:

```

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 58
-----
Process exited after 29.21 seconds with return value 0
Press any key to continue . .

```

Compiler Log:

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\C++\strassen\matrix.exe
- Output Size: 129.962890625 KiB
- Compilation Time: 0.53s

```

16. Merge Sort:

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int leftArr[n1], rightArr[n2];
    for (i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        rightArr[j] = arr[mid + 1 + j];
    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2)
    {
        if (leftArr[i] < rightArr[j])
            arr[k] = leftArr[i];
        else
            arr[k] = rightArr[j];
        i++;
        j++;
        k++;
    }
    while (i < n1)
        arr[k] = leftArr[i];
    while (j < n2)
        arr[k] = rightArr[j];
}
```

```
k = left;
while (i < n1 && j < n2) {
    if (leftArr[i] <= rightArr[j]) {
        arr[k] = leftArr[i];
        i++;
    }
    else {
        arr[k] = rightArr[j];
        j++;
    }
    k++;
}

// Copy the remaining elements of leftArr[], if any
while (i < n1) {
    arr[k] = leftArr[i];
    i++;
    k++;
}
while (j < n2) {
    arr[k] = rightArr[j];
    j++;
    k++;
}
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int n = sizeof(arr) / sizeof(arr[0]);
```

```

mergeSort(arr, 0, n - 1);
for (int i = 0; i < n; i++)
    printf("%d ", arr[i]);
return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ file named 'merge sort.cpp' containing the provided merge sort implementation. The output window shows the sorted array [5, 6, 7, 11, 12, 13]. The compiler results window shows no errors or warnings, indicating a successful compilation.

17. Divide and Conquer Strategy for Finding MAX and MIN Values:

```

#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  
    {  
        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;
}

```

The screenshot shows the Dev-C++ IDE interface. The left pane displays the source code for 'divide and conquer max and min.cpp'. The right pane shows the terminal window where the program is running. The user enters '5' followed by five integers: 1, 4, 2, 10, and 9. The program then outputs the minimum element as 1 and the maximum element as 10. The terminal also shows the compilation log at the bottom.

```

C:\Users\mandl\divide and conquer max and min.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug print prime numbers.cpp divide and conquer max and min.cpp
1 #include<stdio.h>
2 #include<stdio.h>
3 int max, min;
4 int a[100];
5 void maxmin(int i, int j)
6 {
7     int max1, min1, mid;
8     if(i==j)
9     {
10         max = min = a[i];
11     }
12     else
13     {
14         if(i == j-1)
15         {
16             if(a[i] < a[j])
17             {
18                 max = a[j];
19                 min = a[i];
20             }
21             else
22             {
23                 max = a[i];
24                 min = a[j];
25             }
26         }
27     }
28 }
29
30 int main()
31 {
32     int n;
33     printf("Enter the total number of numbers : ");
34     scanf("%d", &n);
35     maxmin(1, n);
36     printf("Minimum element in an array : %d\n", min);
37     printf("Maximum element in an array : %d\n", max);
38     return 0;
39 }

Enter the total number of numbers : 5
Enter the numbers :
1
4
2
10
9
Minimum element in an array : 1
Maximum element in an array : 10
-----
Process exited after 28.6 seconds with return value 0
Press any key to continue . . .

```

18. Prime numbers between Range:

```

#include<stdio.h>
int main()
{
    int a,b,i,j;

```

```
printf("Enter the ranges:");
scanf("%d %d",&a,&b);
for(i=a;i<=b;i++)
{
    int count=0;
    for(j=1;j<=i;j++)
    {
        if(i%j==0)
        {
            count++;
        }
    }
    if(count==2)
    {
        printf("%d ",i);
    }
}
}
```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ file named 'print prime numbers.cpp'. The code prompts the user for two integers, 'a' and 'b', and then iterates through all integers from 'a' to 'b'. For each integer, it counts the number of divisors. If the count is exactly 2, it prints the number as a prime. The output window on the right shows the program's execution. It asks for ranges, takes '100' as input, lists all prime numbers up to 100, and then exits. The status bar at the bottom provides system information like date and time.

```
C:\Users\mandl\C++\print prime numbers.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools ASyle Window Help
C:\Users\mandl\C++\print pri x + v
Project Classes Debug length of string.cpp [*] Untitled2 [*] Untitled3 binary search.cpp
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,i,j;
5     printf("Enter the ranges:");
6     scanf("%d %d",&a,&b);
7     for(i=a;i<=b;i++)
8     {
9         int count=0;
10        for(j=1;j<=i;j++)
11        {
12            if(i%j==0)
13            {
14                count++;
15            }
16        }
17        if(count==2)
18        {
19            printf("%d ",i);
20        }
21    }
22 }
```

Enter the ranges:
100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Process exited after 4.547 seconds with return value 0
Press any key to continue . . .

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\C++\print prime numbers.exe
- Output Size: 128.125 KiB
- Compilation Time: 0.22s

Line: 20 Col: 14 Sel: 0 Lines: 22 Length: 276 Insert Done parsing in 0 seconds

32°C Partly sunny ENG IN 14:12 08-11-2024

19.Knapsack Problem using Greedy Technique:

```
#include<stdio.h>
int main()
{
    float weight[50],profit[50],ratio[50],Totalvalue,temp,capacity,amount;
    int i,j,num;
    printf("Enter number of items :");
    scanf("%d",&num);
    for (i = 0; i < num; i++)
    {
        printf("\n\nEnter Weight and Profit for item[%d] :\n",i);
        scanf("%f %f", &weight[i], &profit[i]);
    }
    printf("\n\nEnter capacity of knapsack :\n");
    scanf("%f",&capacity);

    for(i=0;i<num;i++)
        ratio[i]=profit[i]/weight[i];

    for (i = 0; i < num; i++)
    {
        for (j = i + 1; j < num; j++)
        {
            if (ratio[i] < ratio[j])
            {
                temp = ratio[j];
                ratio[j] = ratio[i];
                ratio[i] = temp;

                temp = weight[j];
                weight[j] = weight[i];
                weight[i] = temp;

                temp = profit[j];
            }
        }
    }
}
```

```
    profit[j] = profit[i];
    profit[i] = temp;
}
}
}

printf("\nKnapsack Problem using Greedy Method :\n");
for (i = 0; i < num; i++)
{
if (weight[i] > capacity)
    break;
else
{
    Totalvalue = Totalvalue + profit[i];
    capacity = capacity - weight[i];
}
}
if (i < num)
    Totalvalue = Totalvalue + (ratio[i]*capacity);
printf("\nThe maximum value is :%f\n",Totalvalue);
return 0;
}
```

```

C:\Users\mandl\Downloads\Knapsack problem using greedy.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals) Project Classes Debug print prime numbers.cpp divide and conquer max and min.cpp knapsack problem
1 #include<stdio.h>
2 int main()
3 {
4     float weight[50], profit[50], ratio[50], TotalProfit = 0;
5     int i, j, num;
6     printf("Enter number of items : ");
7     scanf("%d", &num);
8     for (i = 0; i < num; i++)
9     {
10         printf("\n\nEnter Weight and Profit for item[%d] : ", i+1);
11         scanf("%f %f", &weight[i], &profit[i]);
12     }
13     printf("\n\nEnter capacity of knapsack : ");
14     scanf("%f", &capacity);
15
16     for(i=0;i<num;i++)
17     {
18         ratio[i]=profit[i]/weight[i];
19     }
20
21     for (i = 0; i < num; i++)
22     {
23         for (j = i + 1; j < num; j++)
24         {
25             if (ratio[i] < ratio[j])
26             {
27                 float tempWeight = weight[i];
28                 weight[i] = weight[j];
29                 weight[j] = tempWeight;
30
31                 float tempProfit = profit[i];
32                 profit[i] = profit[j];
33                 profit[j] = tempProfit;
34
35                 ratio[i] = profit[i] / weight[i];
36             }
37         }
38     }
39
40     for (i = 0; i < num; i++)
41     {
42         TotalProfit += profit[i];
43     }
44
45     printf("KnapSack Problem using Greedy Method : \n");
46     printf("The maximum value is : %f\n", TotalProfit);
47
48     return 0;
49 }

```

Enter Weight and Profit for item[1] :
2
5

Enter Weight and Profit for item[2] :
3
10

Enter Weight and Profit for item[3] :
4
9

Enter Weight and Profit for item[4] :
5
15

Enter capacity of knapsack :
10

Knapsack Problem using Greedy Method :

The maximum value is : 30.000000

Process exited after 77.2 seconds with return value 0
Press any key to continue . . .

Line: 55 Col: 2 Sek: 0 Lines: 55 Length: 1409 Insert Done parsing in 0.015 seconds

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\Downloads\Knapsack problem using greedy.exe
- Output Size: 129.3037109375 Kib
- Compilation Time: 0.39s

1222 12-11-2024

20. MST (minimum spanning tree) using Kruskal's Algorithm:

```

#include <stdio.h>
#include <stdlib.h>
typedef struct Edge {
    int src;
    int dest;
    int weight;
} Edge;
typedef struct Graph {
    int V;
    int E;
    Edge* edges;
} Graph;
Graph* createGraph(int V, int E) {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->V = V;
    graph->E = E;
}

```

```
graph->edges = (Edge*)malloc(E * sizeof(Edge));
return graph;
}

typedef struct DSU {
    int* parent;
    int* rank;
} DSU;

DSU* createDSU(int V) {
    DSU* dsu = (DSU*)malloc(sizeof(DSU));
    dsu->parent = (int*)malloc(V * sizeof(int));
    dsu->rank = (int*)malloc(V * sizeof(int));
    for (int i = 0; i < V; i++) {
        dsu->parent[i] = i;
        dsu->rank[i] = 0;
    }
    return dsu;
}

int find(DSU* dsu, int x) {
    if (dsu->parent[x] != x) {
        dsu->parent[x] = find(dsu, dsu->parent[x]);
    }
    return dsu->parent[x];
}

void unionSets(DSU* dsu, int x, int y) {
    int rootX = find(dsu, x);
    int rootY = find(dsu, y);

    if (rootX != rootY) {
        if (dsu->rank[rootX] < dsu->rank[rootY]) {
            dsu->parent[rootX] = rootY;
        } else if (dsu->rank[rootX] > dsu->rank[rootY]) {
            dsu->parent[rootY] = rootX;
        } else {
            dsu->parent[rootY] = rootX;
            dsu->rank[rootX]++;
        }
    }
}
```

```
}

int compareEdges(const void* a, const void* b) {
    Edge* edgeA = (Edge*)a;
    Edge* edgeB = (Edge*)b;
    return edgeA->weight - edgeB->weight;
}

void kruskalMST(Graph* graph) {
    qsort(graph->edges, graph->E, sizeof(Edge), compareEdges);

    DSU* dsu = createDSU(graph->V);

    Edge* result = (Edge*)malloc((graph->V - 1) * sizeof(Edge));
    int edgeCount = 0;
    int i = 0;

    while (edgeCount < graph->V - 1 && i < graph->E) {
        Edge nextEdge = graph->edges[i++];
        int srcRoot = find(dsu, nextEdge.src);
        int destRoot = find(dsu, nextEdge.dest);

        if (srcRoot != destRoot) {
            result[edgeCount++] = nextEdge;
            unionSets(dsu, srcRoot, destRoot);
        }
    }

    printf("Edges in the Minimum Spanning Tree:\n");
    for (int j = 0; j < edgeCount; j++) {
        printf("%d -- %d : %d\n", result[j].src, result[j].dest, result[j].weight);
    }

    free(result);
    free(dsu->parent);
    free(dsu->rank);
    free(dsu);
}
```

```
int main() {
    int V = 4;
    int E = 5;

    Graph* graph = createGraph(V, E);
    graph->edges[0].src = 0;
    graph->edges[0].dest = 1;
    graph->edges[0].weight = 10;

    graph->edges[1].src = 0;
    graph->edges[1].dest = 2;
    graph->edges[1].weight = 6;

    graph->edges[2].src = 0;
    graph->edges[2].dest = 3;
    graph->edges[2].weight = 5;

    graph->edges[3].src = 1;
    graph->edges[3].dest = 3;
    graph->edges[3].weight = 15;

    graph->edges[4].src = 2;
    graph->edges[4].dest = 3;
    graph->edges[4].weight = 4;

    kruskalMST(graph);

    free(graph->edges);
    free(graph);

    return 0;
}
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 typedef struct Edge {
4     int src;
5     int dest;
6     int weight;
7 } Edge;
8 typedef struct Graph {
9     int V;
10    int E;
11    Edge* edges;
12 } Graph;
13 Graph* createGraph(int V, int E) {
14     Graph* graph = (Graph*)malloc(sizeof(Graph));
15     graph->V = V;
16     graph->E = E;
17     graph->edges = (Edge*)malloc(E * sizeof(Edge));
18     return graph;
19 }
20 typedef struct DSU {
21     int* parent;
22     int* rank;
23 } DSU;
24 DSU* createDSU(int V) {

```

Edges in the Minimum Spanning Tree:
2 -- 3 : 4
0 -- 3 : 5
0 -- 1 : 10

Process exited after 1.536 seconds with return value 0
Press any key to continue . . . |

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandl\C++\kruskal's algorithm in MST.exe
- Output Size: 130.516484375 KiB
- Compilation Time: 0.42s

21. OBST (Optimal Binary Search Tree) Technique:

```

#include <stdio.h>
#include <limits.h>
int sum(int freq[], int i, int j);
int optCost(int freq[], int i, int j)
{
    if (j < i)
        return 0;
    if (j == i)
        return freq[i];
    int fsum = sum(freq, i, j);
    int min = INT_MAX;
    for (int r = i; r <= j; ++r)
    {
        int cost = optCost(freq, i, r-1) +
                   optCost(freq, r+1, j);
        if (cost < min)
            min = cost;
    }
}

```

```
    }
    return min + fsum;
}
int optimalSearchTree(int keys[], int freq[], int n)
{
    return optCost(freq, 0, n-1);
}
int sum(int freq[], int i, int j)
{
    int s = 0;
    for (int k = i; k <=j; k++)
        s += freq[k];
    return s;
}
int main()
{
    int keys[] = {10, 12, 20};
    int freq[] = {34, 8, 50};
    int n = sizeof(keys)/sizeof(keys[0]);
    printf("Cost of Optimal BST is %d ",
           optimalSearchTree(keys, freq, n));
    return 0;
}
```

```

C:\Users\mandi\Downloads\OBST technique.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Project Classes Debug print prime numbers.cpp divide and conquer max and min.cpp knapsack prob...
C:\Users\mandi\Downloads\OBST technique.cpp + v
TDM-GCC 4.9.2 64-bit Release
globals
1 #include <stdio.h>
2 #include <limits.h>
3 int sum(int freq[], int i, int j);
4 int optCost(int freq[], int i, int j)
5 {
6     if (j < i)
7         return 0;
8     if (j == i)
9         return freq[i];
10    int fsum = sum(freq, i, j);
11    int min = INT_MAX;
12    for (int r = i; r <= j; ++r)
13    {
14        int cost = optCost(freq, i, r-1) +
15                    optCost(freq, r+1, j);
16        if (cost < min)
17            min = cost;
18    }
19    return min + fsum;
20 }
21 int optimalSearchTree(int keys[], int freq[])
22 {
23     return optCost(freq, 0, n-1);
24 }

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\Downloads\OBST technique.exe
- Output Size: 128.5361328125 KiB
- Compilation Time: 0.39s

Line: 31 Col: 2 Sel: 0 Lines: 40 Length: 904 Insert Done parsing in 0.016 seconds
Search 12:43 12-11-2024 ENG IN

```

22. Binary Coefficient:

```

#include <stdio.h>
int binomialCoeff(int n, int k) {
    if (k > n)
        return 0;
    if (k == 0 || k == n)
        return 1;
    return binomialCoeff(n - 1, k - 1)
        + binomialCoeff(n - 1, k);
}

int main() {
    int n,k;
    printf("Enter the n value:\n");
    scanf("%d",&n);
    printf("Enter the k value:");

```

```

scanf("%d",&k);
printf("%d", binomialCoeff(n, k));
return 0;
}

```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ file named 'Binomial coefficient.cpp' with the following content:

```

1 #include <stdio.h>
2 int binomialCoeff(int n, int k) {
3     if (k > n)
4         return 0;
5     if (k == 0 || k == n)
6         return 1;
7     return binomialCoeff(n - 1, k - 1)
8         + binomialCoeff(n - 1, k);
9 }
10
11 int main() {
12     int n,k;
13     printf("Enter the n value:\n");
14     scanf("%d",&n);
15     printf("Enter the k value:\n");
16     scanf("%d",&k);
17     printf("%d", binomialCoeff(n, k));
18     return 0;
19 }

```

The output window shows the following interaction:

```

Enter the n value:
5
Enter the k value:2
10
Process exited after 5.475 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom shows compilation results:

- Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\C++\Binomial coefficient.exe
- Output Size: 128.83203125 Kib
- Compilation Time: 0.30s

The system tray shows the date and time as 12-11-2024 12:50.

23. Reverse of a Number:

```

#include<stdio.h>
int main()
{
    int n,r,sum=0;
    printf("Enter the number:");
    scanf("%d",&n);
    while(n>0)
    {
        r=n%10;
        sum=sum*10+r;
    }
}

```

```

n=n/10;
}
printf("%reversed number=%d",sum);
}

```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a file named 'reverse a numberr.cpp' with the following content:

```

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

```

On the right, the terminal window shows the execution of the program:

```

Enter the number:12345
reversed number=54321
Process exited after 15.23 seconds with return value 0
Press any key to continue . . .

```

Below the terminal, the compiler results window shows:

- Compilation results...
- Summary:
 - Errors: 0
 - Warnings: 0
 - Output Filename: C:\Users\mandi\Documents\reverse a numberr.cpp
 - Output Size: 128.123046875 KiB
 - Compilation Time: 0.45s

The status bar at the bottom indicates the current line (Line: 3), column (Col: 3), selected text (Sel: 0), total lines (Lines: 14), and total length (Length: 210). The taskbar at the bottom shows various application icons.

24.Perfect Number:

```

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

```

```

if(sum==n)
{
    printf("%d is a perfect number",n);
}
else
{
    printf("%d is not a perfect number",n);
}

```

The screenshot shows the Dev-C++ IDE interface. The left pane displays the source code for a file named 'perfect number.cpp'. The right pane has two windows: one for the terminal showing the execution of the program and another for the compiler showing the results of the compilation.

```

C:\Users\mandi\Documents\Dev-C++\perfect number.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
File Edit View Project Execute Tools AStyle Window Help
(globals) C:\Users\mandi\Documents\Dev-C++\perfect number.cpp
Project Classes Debug reverse a number.cpp perfect number.cpp
1 #include<stdio.h>
2 int main()
3 {
4     int n,sum=0,i;
5     printf("Enter the number:");
6     scanf("%d",&n);
7     for(i=1;i<n;i++)
8     {
9         if(n%i==0)
10        {
11            sum=sum+i;
12        }
13    }
14    if(sum==n)
15    {
16        printf("%d is a perfect number",n);
17    }
18    else
19    {
20        printf("%d is not a perfect number",n);
21    }
22 }

Compiler Resources Compile Log Find Results Close
Compilation results...
Abort Compilation
-----  

- Errors: 0  

- Warnings: 0  

- Output Filename: C:\Users\mandi\Documents\Dev-C++\perfect number.exe  

- Output Size: 128.6015625 KiB  

- Compilation Time: 0.38s

Line: 20 Col: 46 Sel: 0 Lines: 22 Length: 287 Insert Done parsing in 0 seconds
31°C Mostly cloudy Search ENG IN 13:24 09-11-2024

```

25.TSP (Travelling Sales Man Problem) using Dynamic programming:

```

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

#define MAX 16
#define INF INT_MAX

```

```
int dist[MAX][MAX];
int dp[1 << MAX][MAX];

int tsp(int mask, int pos, int n) {
    if (mask == (1 << n) - 1) return dist[pos][0];
    if (dp[mask][pos] != -1) return dp[mask][pos];

    int ans = INF;
    for (int city = 0; city < n; city++) {
        if ((mask & (1 << city)) == 0) {
            int newAns = dist[pos][city] + tsp(mask | (1 << city), city, n);
            if (newAns < ans) ans = newAns;
        }
    }

    return dp[mask][pos] = ans;
}

int main() {
    int n;
    printf("Enter the number of cities: ");
    scanf("%d", &n);

    printf("Enter the distance matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &dist[i][j]);

    for (int i = 0; i < (1 << n); i++)
        for (int j = 0; j < n; j++)
            dp[i][j] = -1;

    printf("The minimum cost to visit all cities is: %d\n", tsp(1, 0, n));
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ file named 'reverse a number.cpp' containing a recursive function for solving the Traveling Salesman Problem (TSP) using dynamic programming. The code includes #include directives for stdio.h and limits.h, defines MAX as 16, and uses a dp array to store intermediate results. The main function 'tsp' takes a mask (representing cities visited) and a current city index 'pos'. It returns the minimum cost to visit all cities from the current city. The function uses a loop to iterate through cities and update the minimum cost if a better path is found. The code also handles base cases where all cities have been visited.

On the right, a terminal window titled 'C:\Users\mandl\TSP user' shows the execution of the program. It prompts the user for the number of cities (4) and then asks for the distance matrix. The matrix is provided as:

```
0 10 15 20
10 0 35 25
15 35 0 30
20 25 30 0
```

The program outputs the minimum cost to visit all cities, which is 80. It also prints the execution time (44.07 seconds) and a message to press any key to continue.

At the bottom, the status bar shows the line, column, and character counts (Line: 42, Col: 1, Sel: 0), the search bar, and various system icons.

26. Print the pattern 1

```
1 2
1 2 3
1 2 3 4
```

```
#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=4;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("%d ",j);
        }
        printf("\n");
    }
}
```

The screenshot shows the Dev-C++ IDE interface. The project navigation bar at the top lists 'print prime numbers.cpp' and 'divide by 2.cpp'. The main code editor window displays the following C++ code:

```

1 #include<stdio.h>
2 int main()
3 {
4     int i,j;
5     for(i=1;i<=4;i++)
6     {
7         for(j=1;j<=i;j++)
8         {
9             printf("%d ",j);
10        }
11    printf("\n");
12 }

```

The output window shows the printed pattern:

```

1
1 2
1 2 3
1 2 3 4

```

Below the output, the message "Process exited after 10.04 seconds with return value 0" is displayed, followed by "Press any key to continue . . .".

The bottom status bar shows the current line (Line: 12), column (Col: 6), and other details like compilation results and system status.

27. Floyd's Warshall Algorithm:

```

#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;
}

```

The screenshot shows the Dev-C++ IDE interface with two windows open. The left window displays the source code for a C++ program named 'floyedes.cpp'. The right window shows the terminal output of the program's execution.

Code (floyedes.cpp):

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void floydWarshall(int **graph, int n)
5 {
6     int i, j, k;
7     for (k = 0; k < n; k++)
8     {
9         for (i = 0; i < n; i++)
10        {
11            for (j = 0; j < n; j++)
12            {
13                if (graph[i][j] > graph[i][k] + graph[k][j])
14                    graph[i][j] = graph[i][k] + graph[k][j];
15            }
16        }
17    }
18
19    int main(void)
20    {
21        int n, i, j;
22        printf("Enter the number of vertices: ");
23    }

```

Terminal Output:

```

Enter the number of vertices: 4
Enter the edges:
[0][0]: 0
[0][1]: 100
[0][2]: 3
[0][3]: 100
[1][0]: 2
[1][1]: 0
[1][2]: 100
[1][3]: 100
[2][0]: 100
[2][1]: 7
[2][2]: 0
[2][3]: 1
[3][0]: 6
[3][1]: 100
[3][2]: 100
[3][3]: 0
The original graph is:
0 100 3 100
2 0 100 100
100 7 0 1
6 100 100 0
The shortest path matrix is:
0 10 3 4
2 0 5 6
7 7 0 1
6 16 9 0

```

The terminal output shows the user inputting 4 for the number of vertices, followed by the edges: (0,1) = 100, (0,2) = 3, (0,3) = 100, (1,0) = 2, (1,2) = 100, (1,3) = 100, (2,0) = 100, (2,1) = 7, (2,3) = 1, (3,0) = 6, (3,1) = 100, (3,2) = 100, (3,3) = 0. The program then prints the original graph and the resulting shortest path matrix.

28.Pascal's Triangle:

```
#include <stdio.h>
void printPascal(int n)
{
    for (int line = 1; line <= n; line++) {
        for (int space = 1; space <= n - line; space++)
            printf(" ");
        int coef = 1;
        for (int i = 1; i <= line; i++) {
            printf("%4d", coef);
            coef = coef * (line - i) / i;
        }
        printf("\n");
    }
}
int main()
{
    int n = 5;
    printPascal(n);
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The main window displays a C++ code editor with the file 'printPascal.cpp' open. The code prints a Pascal's triangle for n=5. The output window shows the triangle:

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
  -----
Process exited after 10.05 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom indicates the compilation results:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\mandi\C++\pascal's triangle.exe
- Output Size: 128.6591796875 KiB
- Compilation Time: 0.41s

29. Sum of Digits of a Number:

```

#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n,r,sum=0;
    printf("Enter the number:");
    scanf("%d", &n);
    n=abs(n);
    while(n>0)
    {
        r=n%10;
        sum=sum+r;
        n=n/10;
    }
    printf("%d",sum);
}

```

```
    return 0;
}
```

The screenshot shows the Dev-C++ IDE interface. The code editor displays a C++ program named 'sumofdigits.cpp' which calculates the sum of digits of a given number. The output window shows the program's execution, including user input '12345', the calculated sum '15', and the exit message. The compiler results window shows no errors or warnings, indicating a successful compilation.

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 int main()
4 {
5     int n,r,sum=0;
6     printf("Enter the number:");
7     scanf("%d", &n);
8     n=abs(n);
9     while(n>0)
10    {
11        r=n%10;
12        sum=sum+r;
13        n=n/10;
14    }
15    printf("%d",sum);
16    return 0;
17 }
```

30.Insert an Element in an Array:

```
#include<stdio.h>
int main()
{
    int a[]={1,2,3,4,5};
    int l,t,i;
    l=sizeof(a)/sizeof(a[0]);
    printf("Enter the element to insert:");
    scanf("%d",&t);
    a[l]=t;
    for(i=0;i<=l;i++)
    {
        printf("%d ",a[i]);
    }
```

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays a C++ program named 'insert element.cpp'. The code defines an array 'a' with elements 1, 2, 3, 4, 5. It prompts the user to enter an element to insert at index 1. The user enters '6'. The program then prints the modified array: 1 2 3 4 5 6. Below the code editor is the 'Compiler' tab of the IDE, showing compilation results: Errors: 0, Warnings: 0, Output Filename: C:\Users\mand1\C++\insert e, Output Size: 128.1015625 KiB, Compilation Time: 0.25s. To the right of the IDE is a terminal window titled 'C:\Users\mand1\C++\insert el'. It shows the user input 'Enter the element to insert:6' followed by the output '1 2 3 4 5 6'. The terminal also displays the message 'Process exited after 2.974 seconds with return value 0' and 'Press any key to continue . . .' At the bottom of the screen is the Windows taskbar.

```

1 #include<stdio.h>
2 int main()
3 {
4     int a[]={1,2,3,4,5};
5     int l,i;
6     l=sizeof(a)/sizeof(a[0]);
7     printf("Enter the element to insert:");
8     scanf("%d",&t);
9     a[1]=t;
10    for(i=0;i<l;i++)
11    {
12        printf("%d ",a[i]);
13    }
14 }

```

31. Sum of Subsets Using Backtracking:

```

#include <stdio.h>
void subsetSum(int arr[], int n, int target_sum, int index, int current_sum,
int current_subset[], int subset_size) {
    if (current_sum == target_sum) {
        printf("{ ");
        for (int i = 0; i<subset_size; i++) {
            printf("%d ", current_subset[i]);
        }
        printf("}\n");
        return;
    }
    if (current_sum>target_sum || index == n) {
        return;
    }
    current_subset[subset_size] = arr[index];

```

```

subsetSum(arr, n, target_sum, index + 1, current_sum + arr[index],
          current_subset, subset_size + 1);
subsetSum(arr, n, target_sum, index + 1, current_sum, current_subset,
          subset_size);
}
void findAllSubsets(int arr[], int n, int target_sum) {
    int current_subset[n];
    subsetSum(arr, n, target_sum, 0, 0, current_subset, 0);
}
int main() {
    int arr[] = {10, 7, 5, 18, 12, 20, 15};
    int target_sum = 35;
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Subsets with sum %d are:\n", target_sum);
    findAllSubsets(arr, n, target_sum);
    return 0;
}

```

OUTPUT:

```

Subsets with sum 35 are:
{ 10 7 18 }
{ 10 5 20 }
{ 5 18 12 }
{ 20 15 }

-----
Process exited after 0.0709 seconds with return value 0
Press any key to continue . . .

```

32.GRAPH COLOURING USING BACKTRACKING:

```

#include <stdio.h>
#include <stdbool.h>
#define N 4
bool isSafe(int vertex, int graph[N][N], int colors[], int color) {
    for (int i = 0; i < N; i++) {
        if (graph[vertex][i] && colors[i] == color) {
            return false;
        }
    }
    return true;
}

int solve(int graph[N][N], int colors[], int vertex) {
    if (vertex == N) {
        return true;
    }
    for (int color = 1; color <= N; color++) {
        if (isSafe(vertex, graph, colors, color)) {
            colors[vertex] = color;
            if (solve(graph, colors, vertex + 1)) {
                return true;
            }
            colors[vertex] = 0;
        }
    }
    return false;
}

int main() {
    int graph[N][N] = {{0, 1, 1, 1}, {1, 0, 1, 1}, {1, 1, 0, 1}, {1, 1, 1, 0}};
    int colors[N];
    if (solve(graph, colors, 0)) {
        for (int i = 0; i < N; i++) {
            printf("%d ", colors[i]);
        }
    } else {
        printf("No solution exists.");
    }
    return 0;
}

```

```
        }
    }
    return true;
}
bool graphColoring(int graph[N][N], int m, int colors[], int vertex) {
    if (vertex == N) {
        return true;
    }
    for (int color = 1; color <= m; color++) {
        if (isSafe(vertex, graph, colors, color)) {
            colors[vertex] = color;
            if (graphColoring(graph, m, colors, vertex + 1)) {
                return true;
            }
        }
    }
    colors[vertex] = 0;
    return false;
}
void solveGraphColoring(int graph[N][N], int m) {
    int colors[N] = {0};
    if (graphColoring(graph, m, colors, 0)) {
        printf("Solution found:\n");
        for (int i = 0; i < N; i++) {
            printf("Vertex %d -> Color %d\n", i, colors[i]);
        }
    } else {
        printf("No solution exists\n");
    }
}
int main() {
    int graph[N][N] = {
        {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
    };
}
```

```
    int m = 3;
    solveGraphColoring(graph, m);
    return 0;
}
```

OUTPUT:

```
Solution found:
Vertex 0 -> Color 1
Vertex 1 -> Color 2
Vertex 2 -> Color 3
Vertex 3 -> Color 2

-----
Process exited after 0.06214 seconds with return value 0
Press any key to continue . . . |
```

33.CONTAINER LOADING PROBLEM:

```
#include <stdio.h>
int maxLoad = 0;
void backtrack(int weights[], int n, int capacity, int index, int currentLoad)
{
    if (currentLoad > capacity) {
        return;
    }
    if (currentLoad > maxLoad) {
        maxLoad = currentLoad;
    }
    if (index == n) {
        return;
    }
    backtrack(weights, n, capacity, index + 1, currentLoad + weights[index]);
    backtrack(weights, n, capacity, index + 1, currentLoad);
}
int maxContainerLoad(int weights[], int n, int capacity) {
    maxLoad = 0;
    backtrack(weights, n, capacity, 0, 0);
    return maxLoad;
}
int main() {
```

```
int weights[] = {10, 20, 30, 40};  
int n = sizeof(weights) / sizeof(weights[0]);  
int capacity = 50;  
int maxLoadPossible = maxContainerLoad(weights, n, capacity);  
printf("Maximum load that can be loaded: %d\n", maxLoadPossible);  
return 0;  
}
```

OUTPUT:

```
Maximum load that can be loaded: 50  
-----  
Process exited after 0.06523 seconds with return value 0  
Press any key to continue . . . |
```

34.LIST OF ALL FACTORS FOR N VALUE:

```
#include <stdio.h>  
#include <math.h>  
void findFactors(int n) {  
    printf("Factors of %d are:\n", n);  
    for (int i = 1; i <= sqrt(n); i++) {  
        if (n % i == 0) {  
            printf("%d ", i);  
            if (i != n / i) {  
                printf("%d ", n / i);  
            }  
        }  
    }  
    printf("\n");  
}  
int main() {  
    int n;  
    printf("Enter a number to find its factors: ");  
    scanf("%d", &n);  
    findFactors(n);  
    return 0;
```

```
}
```

OUTPUT:

```
Enter a number to find its factors: 6
Factors of 6 are:
1 6 2 3

-----
Process exited after 2.281 seconds with return value 0
Press any key to continue . . . |
```

35.JOB ASSIGNMENT PROBLEM USING BRANCH AND BOUND:

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define N 4
typedef struct Node {
    int cost;
    int lowerBound;
    int jobAssignment[N];
    bool assigned[N];
    int level;
} Node;
int calculateLowerBound(int costMatrix[N][N], bool assigned[N], int level)
{
    int lowerBound = 0;

    for (int i = level; i < N; i++) {
        int minCost = INT_MAX;
        for (int j = 0; j < N; j++) {
            if (!assigned[j] && costMatrix[i][j] < minCost) {
                minCost = costMatrix[i][j];
            }
        }
        lowerBound += minCost;
    }
}
```

```
    }
    return lowerBound;
}

void branchAndBound(int costMatrix[N][N]) {
    int minCost = INT_MAX;
    Node bestNode;
    Node root;
    root.cost = 0;
    root.level = 0;
    for (int i = 0; i < N; i++) {
        root.assigned[i] = false;
        root.jobAssignment[i] = -1;
    }
    root.lowerBound = calculateLowerBound(costMatrix, root.assigned,
    root.level);

    Node queue[N * N];
    int queueSize = 0;
    queue[queueSize++] = root;
    while (queueSize > 0) {
        Node currentNode = queue[--queueSize];
        if (currentNode.lowerBound >= minCost) continue;
        if (currentNode.level == N) {
            if (currentNode.cost < minCost) {
                minCost = currentNode.cost;
                bestNode = currentNode;
            }
            continue;
        }
        for (int job = 0; job < N; job++) {
            if (!currentNode.assigned[job]) {
                Node newNode = currentNode;
                newNode.level++;
                newNode.jobAssignment[currentNode.level - 1] = job;
                newNode.cost += costMatrix[currentNode.level - 1][job];
                newNode.assigned[job] = true;
                newNode.lowerBound = newNode.cost +
                    calculateLowerBound(costMatrix, newNode.assigned, newNode.level);
            }
        }
    }
}
```

```

        if (newNode.lowerBound<minCost) {
            queue[queueSize++] = newNode;
        }
    }
}
printf("Minimum cost: %d\n", minCost);
printf("Job assignments:\n");
for (int i = 0; i< N; i++) {
printf("Person %d -> Job %d\n", i, bestNode.jobAssignment[i]);
}
int main() {
    int costMatrix[N][N] = {
        {9, 2, 7, 8},
        {6, 4, 3, 7},
        {5, 8, 1, 8},
        {7, 6, 9, 4}
    };
    branchAndBound(costMatrix);
    return 0;
}

```

OUTPUT:

```

Minimum cost: 10
Job assignments:
Person 0 -> Job 1
Person 1 -> Job 2
Person 2 -> Job 0
Person 3 -> Job -1

-----
Process exited after 0.04755 seconds with return value 0
Press any key to continue . . .

```

36.LINEAR SEARCH:

```

#include <stdio.h>
int linearSearch(int arr[], int n, int target) {
    for (int i = 0; i< n; i++) {

```

```
if (arr[i] == target) {
    return i;
}
return -1;
}
int main() {
    int arr[] = {34, 21, 56, 78, 90, 23, 12};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 78;
    int result = linearSearch(arr, n, target);
    if (result != -1) {
        printf("Element found at index %d\n", result);
    } else {
        printf("Element not found in the array\n");
    }
    return 0;
}
```

OUTPUT:

```
Element found at index 3
-----
Process exited after 0.06744 seconds with return value 0
Press any key to continue . . . |
```

37. HAMILTONIAN CIRCUIT USING BACKTRACKING:

```
#include <stdio.h>
#include <stdbool.h>
#define V 5
bool canAddToPath(int v, int graph[V][V], int path[], int position) {
    if (graph[path[position - 1]][v] == 0)
        return false;
    for (int i = 0; i < position; i++) {
        if (path[i] == v)
```

```
        return false;
    }
    return true;
}
bool hamiltonianCycle(int graph[V][V], int path[], int position) {
    if (position == V) {
        if (graph[path[position - 1]][path[0]] == 1)
            return true;
        else
            return false;
    }
    for (int v = 1; v < V; v++) {
        if (canAddToPath(v, graph, path, position)) {
            path[position] = v;
            if (hamiltonianCycle(graph, path, position + 1))
                return true;
            path[position] = -1;
        }
    }
    return false;
}
int main() {
    int graph[V][V] = {
        {0, 1, 0, 1, 0},
        {1, 0, 1, 1, 0},
        {0, 1, 0, 1, 1},
        {1, 1, 1, 0, 1},
        {0, 0, 1, 1, 0}
    };
    int path[V];
    for (int i = 0; i < V; i++) {
        path[i] = -1;
    }
    path[0] = 0;
    if (hamiltonianCycle(graph, path, 1)) {
        printf("Hamiltonian Cycle found: \n");
        for (int i = 0; i < V; i++) {
```

```
    printf("%d ", path[i]);
}
printf("%d\n", path[0]);
} else {
printf("No Hamiltonian Cycle found\n");
}
return 0;
}
```

OUTPUT:

```
Hamiltonian Cycle found:  
0 1 2 4 3 0  
-----  
Process exited after 0.05161 seconds with return value 0  
Press any key to continue . . . |
```

38.N QUEENS PROBLEM:

```
#include <stdio.h>
#include <stdbool.h>
#define N 8
int board[N][N];
void printSolution() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (board[i][j] == 1)
                printf(" Q ");
            else
                printf(" . ");
        }
        printf("\n");
    }
    printf("\n");
}
bool isSafe(int row, int col) {
    for (int i = 0; i < row; i++) {
```

```
if (board[i][col] == 1)
    return false;
}
for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
    if (board[i][j] == 1)
        return false;
}
for (int i = row, j = col; i >= 0 && j < N; i--, j++) {
    if (board[i][j] == 1)
        return false;
}
return true;
}
bool solveNQueens(int row) {
    if (row == N)
        return true;
    for (int col = 0; col < N; col++) {
        if (isSafe(row, col)) {
            board[row][col] = 1;
            if (solveNQueens(row + 1))
                return true;
            board[row][col] = 0;
        }
    }
    return false;
}
int main() {
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            board[i][j] = 0;
    if (solveNQueens(0)) {
        printSolution();
    } else {
        printf("No solution exists\n");
    }
    return 0;
}
```

OUTPUT:

```
Q . . . . .
. . . . Q . .
. . . . . . Q
. . . . . Q . .
. . . Q . . .
. . . . . . Q .
. Q . . . .
. . . Q . . .

-----
Process exited after 0.03961 seconds with return value 0
Press any key to continue . . . |
```

39.OPTIMAL COST BY USING APPROPRIATE ALGORITHM:

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define V 5
#define INF INT_MAX
void dijkstra(int graph[V][V], int src) {
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++) {
        dist[i] = INF;
        sptSet[i] = false;
    }
    dist[src] = 0;
    for (int count = 0; count < V - 1; count++) {
        int u = -1;
        for (int v = 0; v < V; v++) {
            if (!sptSet[v] && (u == -1 || dist[v] < dist[u])) {
                u = v;
            }
        }
        sptSet[u] = true;
        for (int v = 0; v < V; v++) {
```

```

        if (graph[u][v] && !sptSet[v] && dist[u] != INF && dist[u] +
graph[u][v] < dist[v]) {
dist[v] = dist[u] + graph[u][v];
}
}
printf("Vertex\tDistance from Source\n");
for (int i = 0; i < V; i++) {
printf("%d\t%d\n", i, dist[i]);
}
}

int main() {
int graph[V][V] = {
{0, 10, 0, 30, 0},
{10, 0, 50, 0, 0},
{0, 50, 0, 20, 10},
{30, 0, 20, 0, 60},
{0, 0, 10, 60, 0}
};

dijkstra(graph, 0);

return 0;
}

```

OUTPUT:

```

Vertex  Distance from Source
0      0
1      10
2      50
3      30
4      60

-----
Process exited after 0.04987 seconds with return value 0
Press any key to continue . . .

```

40. MIN MAX VALUE SEPERATELY FOR ALL NUMBERS IN THE LIST:

```
#include <stdio.h>
void findMinMax(int numbers[], int size, int* min, int* max) {
    *min = numbers[0];
    *max = numbers[0];
    for (int i = 1; i < size; i++) {
        if (numbers[i] < *min) {
            *min = numbers[i];
        }
        if (numbers[i] > *max) {
            *max = numbers[i];
        }
    }
}
int main() {
    int numbers[] = {34, 21, 56, 78, 90, 23, 12};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int min, max;
    findMinMax(numbers, size, &min, &max);
    printf("Minimum value: %d\n", min);
    printf("Maximum value: %d\n", max);
    return 0;
}
```

OUTPUT:

```
Minimum value: 12
Maximum value: 90
-----
Process exited after 0.07009 seconds with return value 0
Press any key to continue . . . |
```