**CSA0683 -DESIGN AND ANALYSIS OF ALGORITHMS**

**(LAB PRACTICAL DAY-3)**

AJAY TANGUTURI

192125078

23)Write a program to find the reverse of a given number.

**Input:**

#include <stdio.h>

int main() {

int n, reverse = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while (n != 0) {

remainder = n % 10;

reverse = reverse \* 10 + remainder;

n /= 10;

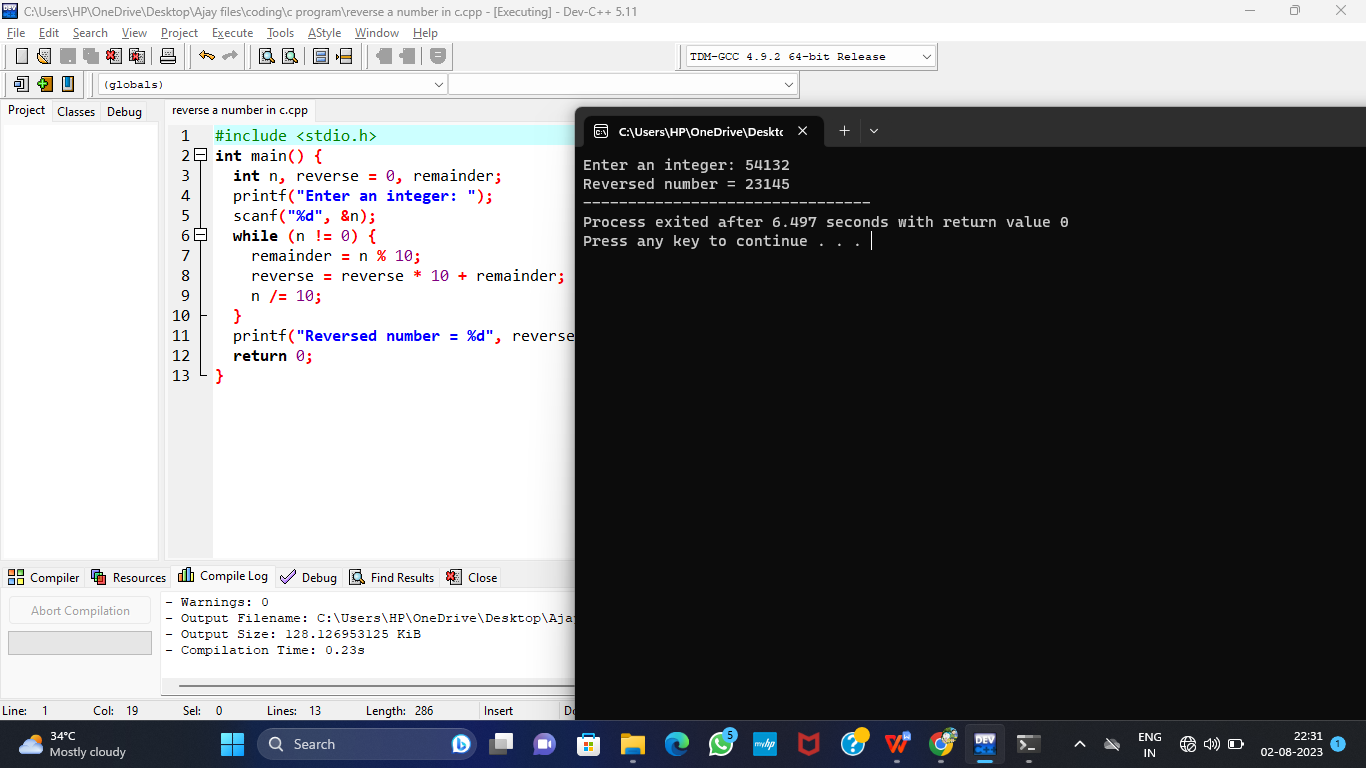
}

printf("Reversed number = %d", reverse);

return 0;

}

**Output:**



1. Write a program to find the perfect number.

**Input:**

#include<stdio.h>

#include<conio.h>

int main()

{

int num, rem, sum = 0, i;

printf("Enter a number\n");

scanf("%d", &num);

for(i = 1; i < num; i++)

{

rem = num % i;

if (rem == 0)

{

sum = sum + i;

}

}

if (sum == num)

printf("is a Perfect Number");

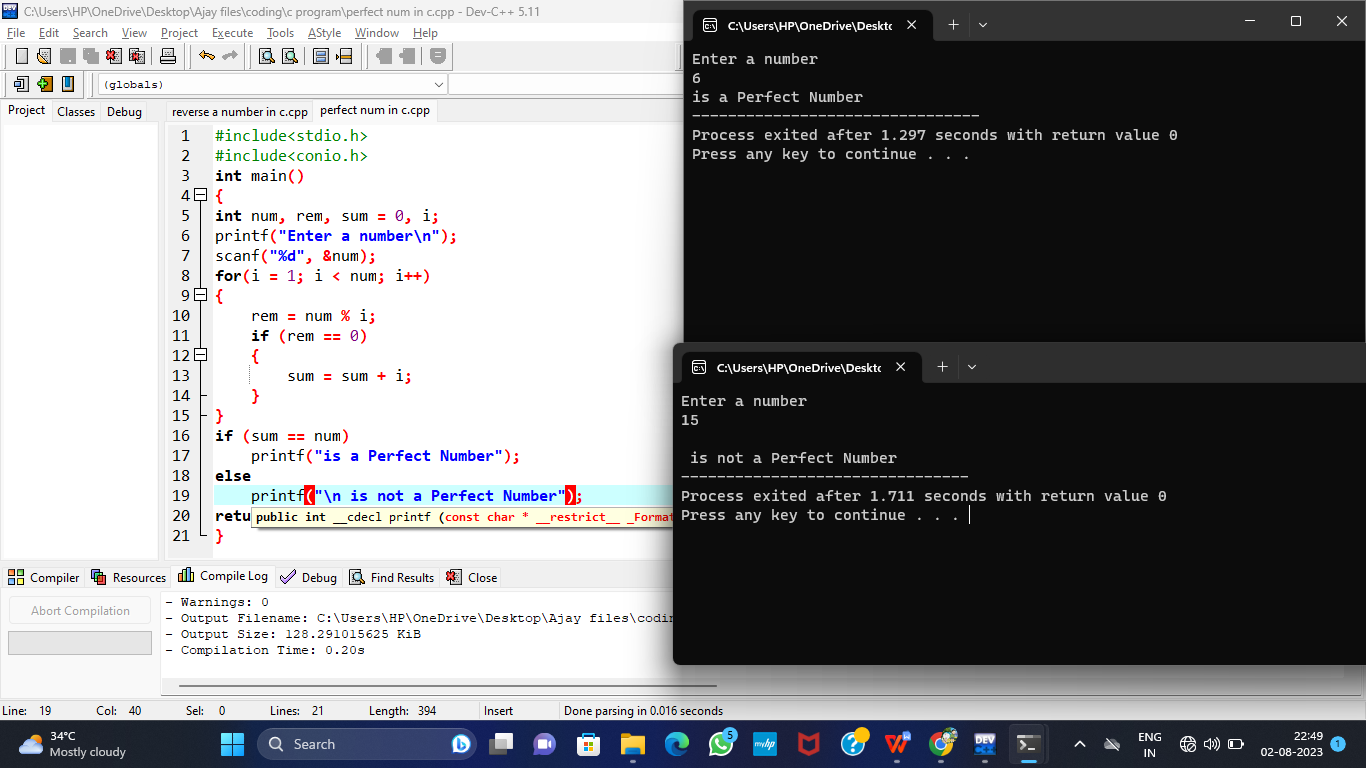
else

printf("\n is not a Perfect Number");

return 0;

}

**Output:**



1. Write a program to perform travelling salesman problem using dynamic programming

**Input:**

#include <stdio.h>

int matrix[25][25], visited\_cities[10], limit, cost = 0;

int tsp(int c)

{

int count, nearest\_city = 999;

int minimum = 999, temp;

for(count = 0; count < limit; count++)

{

if((matrix[c][count] != 0) && (visited\_cities[count] == 0))

{

if(matrix[c][count] < minimum)

{

minimum = matrix[count][0] + matrix[c][count];

}

temp = matrix[c][count];

nearest\_city = count;

}

}

if(minimum != 999)

{

cost = cost + temp;

}

return nearest\_city;

}

void minimum\_cost(int city)

{

int nearest\_city;

visited\_cities[city] = 1;

printf("%d ", city + 1);

nearest\_city = tsp(city);

if(nearest\_city == 999)

{

nearest\_city = 0;

printf("%d", nearest\_city + 1);

cost = cost + matrix[city][nearest\_city];

return;

}

minimum\_cost(nearest\_city);

}

int main()

{

int i, j;

printf("Enter Total Number of Cities:\t");

scanf("%d", &limit);

printf("\nEnter Cost Matrix\n");

for(i = 0; i < limit; i++)

{

printf("\nEnter %d Elements in Row[%d]\n", limit, i + 1);

for(j = 0; j < limit; j++)

{

scanf("%d", &matrix[i][j]);

}

visited\_cities[i] = 0;

}

printf("\nEntered Cost Matrix\n");

for(i = 0; i < limit; i++)

{

printf("\n");

for(j = 0; j < limit; j++)

{

printf("%d ", matrix[i][j]);

}

}

printf("\n\nPath:\t");

minimum\_cost(0);

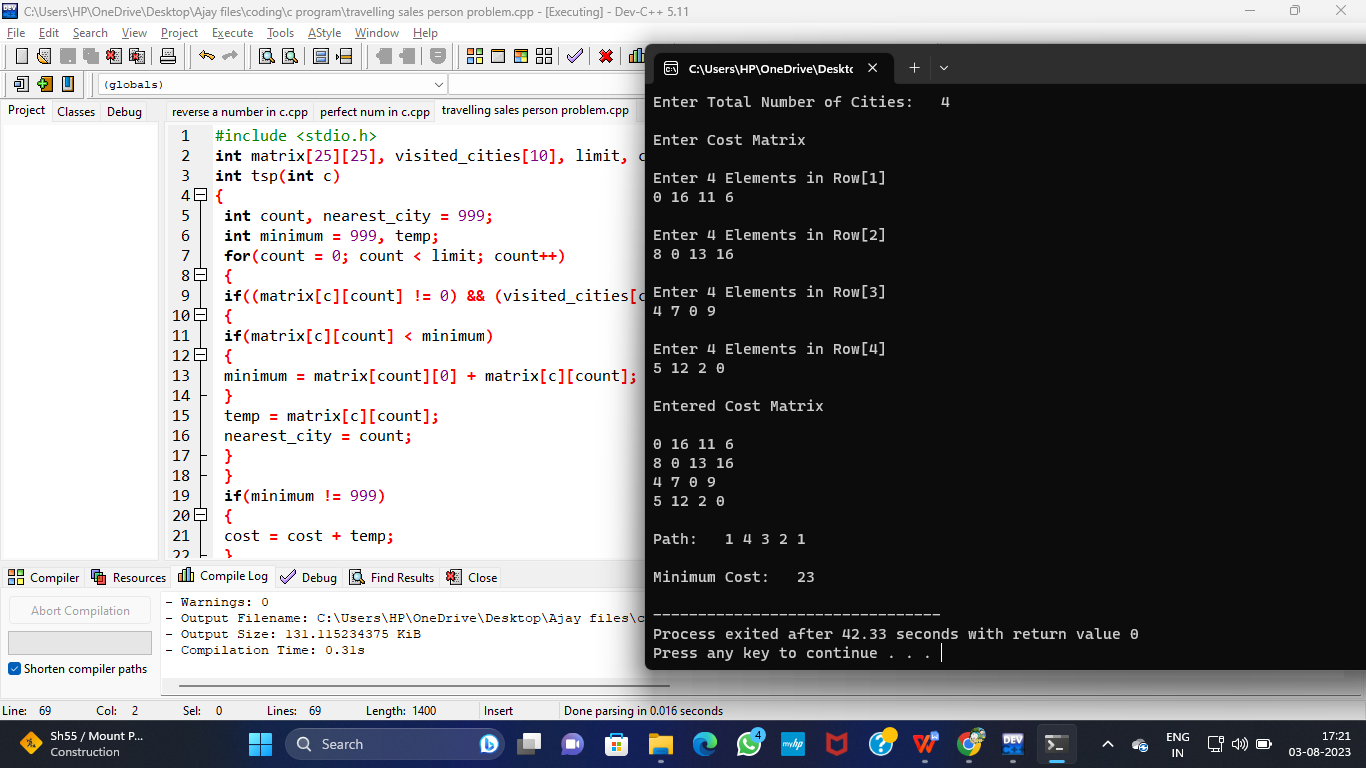
printf("\n\nMinimum Cost: \t");

printf("%d\n", cost);

return 0;

}

**Output:**



26)Write a program for the given pattern

If n=4

1

1 2

1 2 3

1 2 3 4

**Input:**

#include <stdio.h>

int main()

{

int i,j,n;

printf("enter the value);

scanf("%d",&n);

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

{

for(j=1;j<=i;j++)

{

printf("%d",j);

}

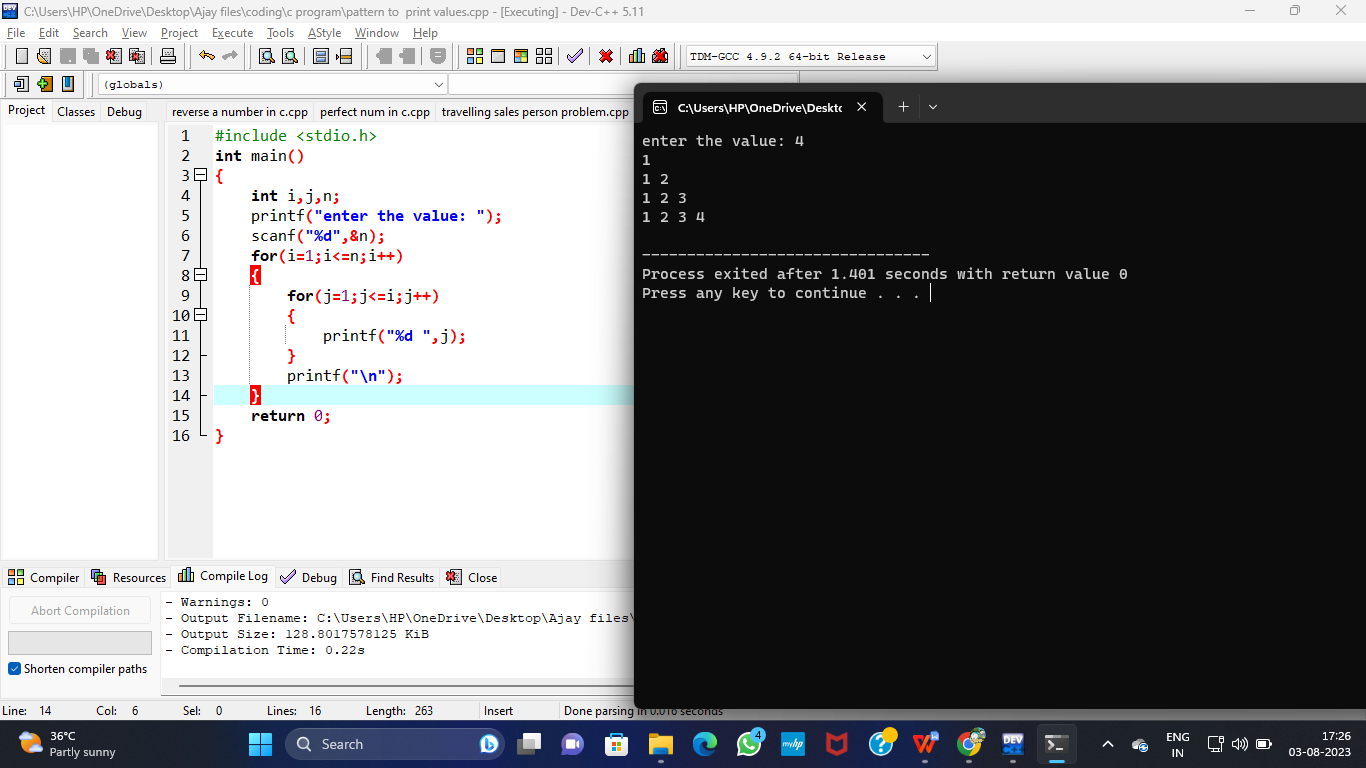
printf("\n");

}

return 0;

}

**Output:**



27)Write a program to perform Floyd’s algorithm

**Input:**

#include<stdio.h>

#include<cstdlib>

int 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]);

}

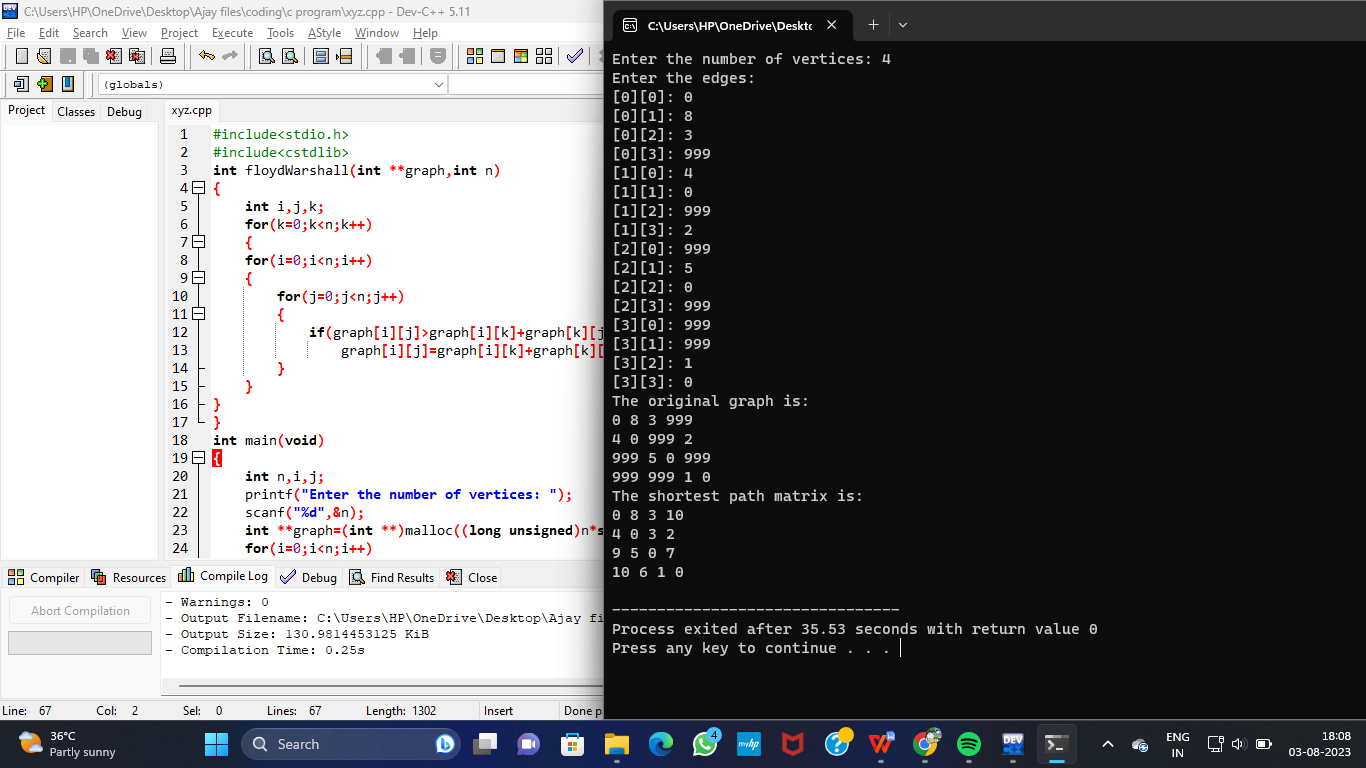
printf("\n");

}

return 0;

}

**Output:**



1. Write a program for pascal triangle.

**Input:**

#include <stdio.h>

int fact(int n)

{

int a;

for(a=1;n>1;n--)

a\*= n;

return a;

}

int combination(int n,int r)

{

return fact(n)/(fact(n-r)\*fact(r));

}

int main()

{

int rows;

int i,j;

printf("Enter Number of Rows: ");

scanf("%d", &rows);

for(i=0;i<=rows;i++)

{

for(j=0;j<=rows-i; j++)

printf(" ");

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

printf(" %3d",combination(i, j));

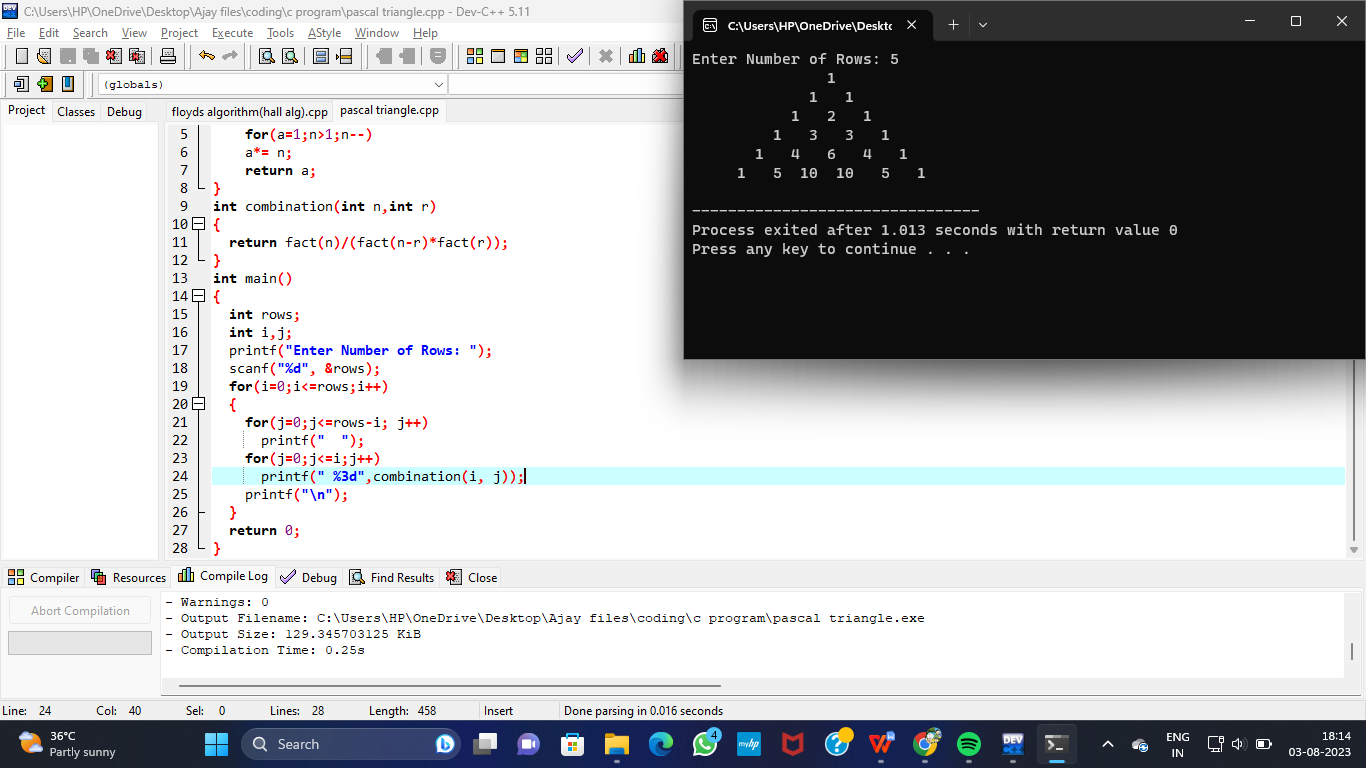
printf("\n");

}

return 0;

}

**Output:**



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

**Input:**

#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];

int main() {

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;

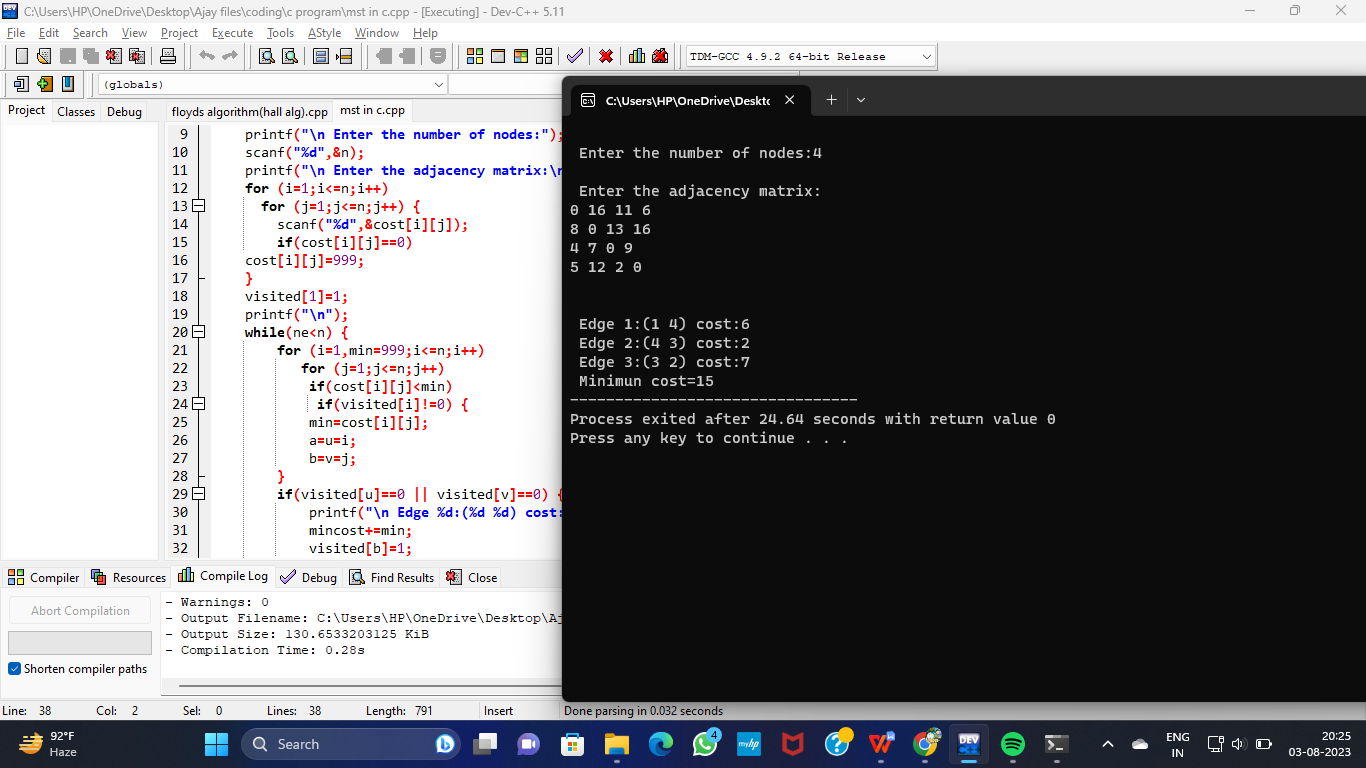
}

printf("\n Minimun cost=%d",mincost);

return 0;

}

**Output:**



30)Write a program to find the sum of digits.

**Input:**

#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;

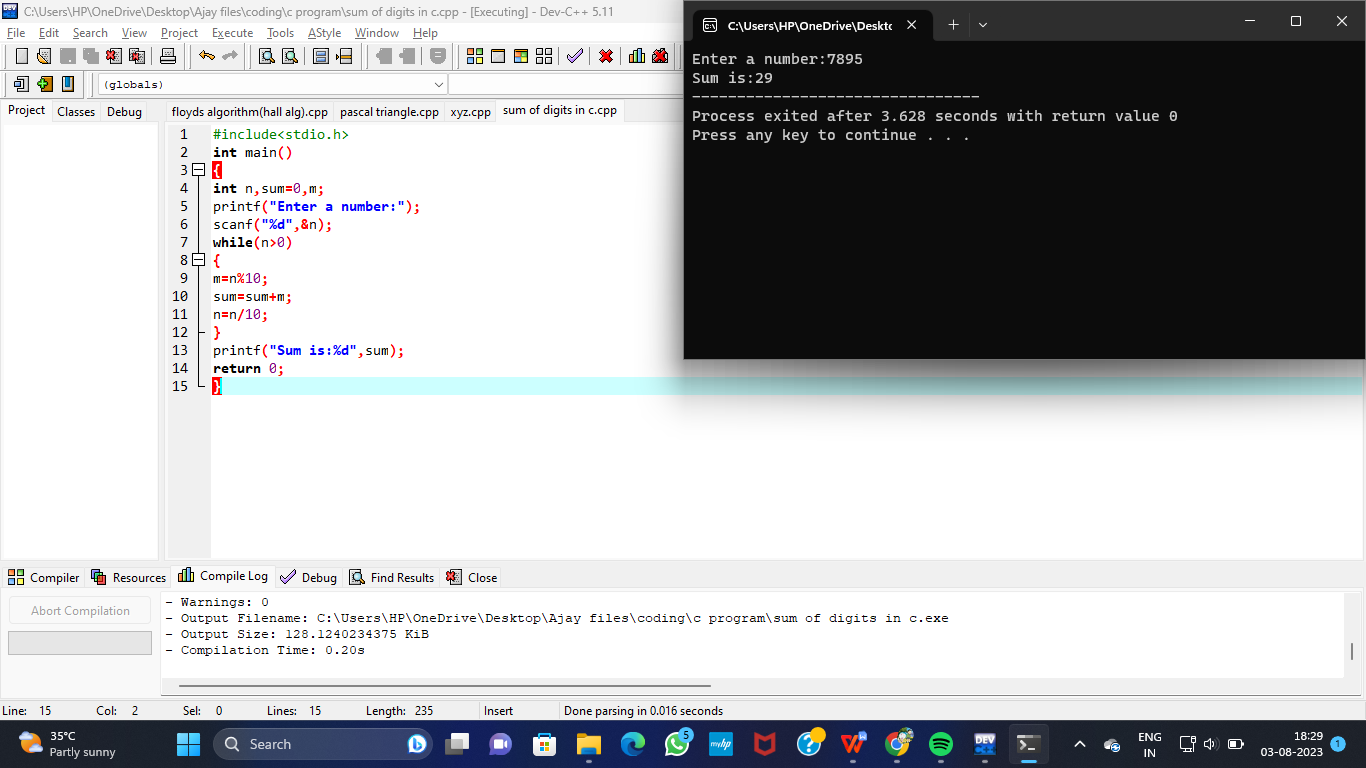
}

printf("Sum is:%d",sum);

return 0;

}

**Output:**



31)Write a program to print minimum and maximum value sequency for all the numbers in a list.

**Input:**

#include<stdio.h>

int main()

{

int array[100],n,c,d,position,t,min,max,z;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter %d integers\n", n);

z=n-1;

for(c=0;c<n;c++)

{

scanf("%d",&array[c]);

}

for(c=0;c<(n-1);c++)

{

position=c;

for(d=c+1;d<n;d++)

{

if(array[position]>array[d])

position=d;

}

if(position!=c)

{

t=array[c];

array[c]=array[position];

array[position]=t;

}

}

printf("\n Sorted list in ascending order:\n");

for(c=0;c<n;c++)

{

printf("%d,",array[c]);

}

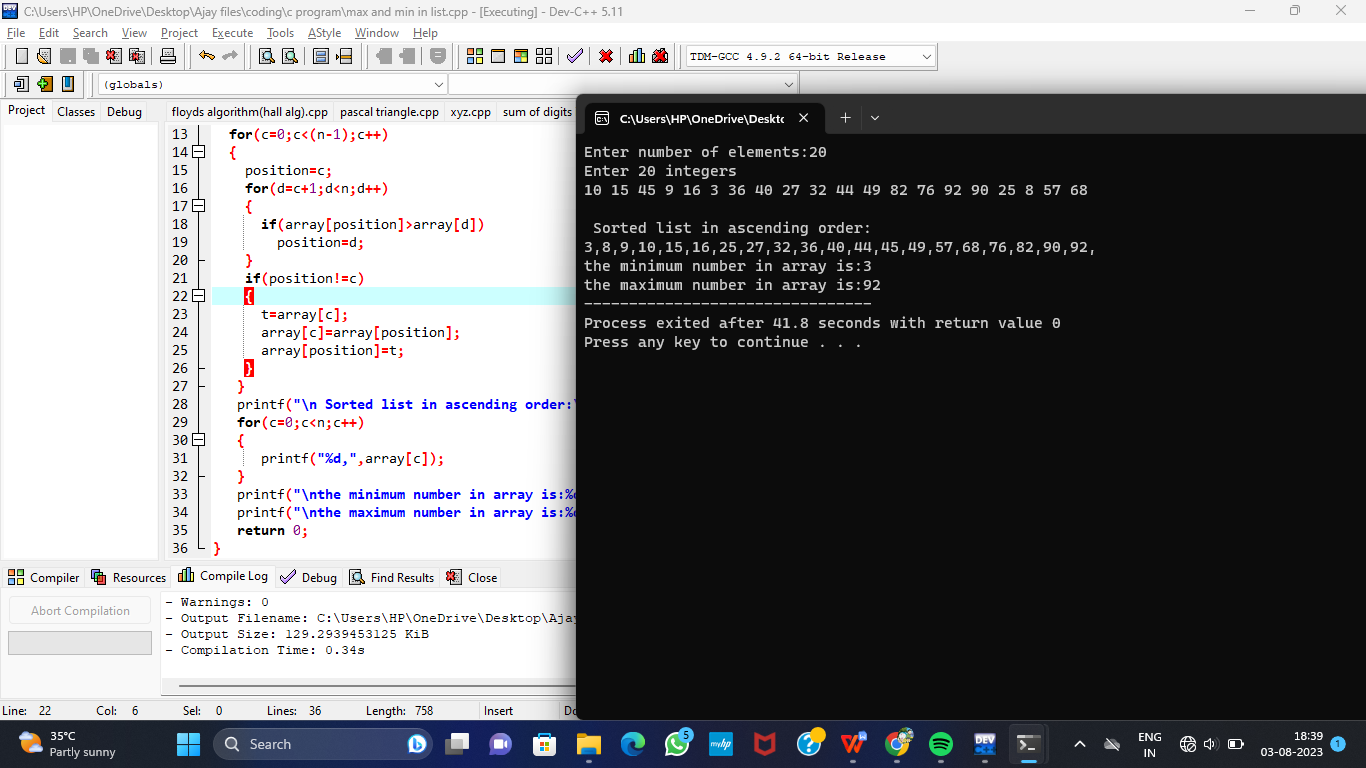
printf("\nthe minimum number in array is:%d",array[0]);

printf("\nthe maximum number in array is:%d",array[z]);

return 0;

}

**Output:**



32)Write a program to perform n Queens Problem using backtracking.

**Input:**

#include<stdio.h>

#include<math.h>

int board[20],count;

int main()

{

int n,i,j;

void queen(int row,int n);

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:**

