1.N Queen problem:

Program:

#define N 4

#include<stdio.h>

#include <stdbool.h>

int count=0;

void printSolution(int board[N][N])

{count++;

for (int i = 0; i < N; i++)

{count++;

for (int j = 0; j < N; j++)

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

printf("\n");

}count++;

}

bool isSafe(int board[N][N], int row, int col)

{

int i, j;

count++;

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

if (board[row][i])

return false;

count++;

for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return false;

count++;

for (i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return false;

count++;

return true;

count++;

}

bool solveNQUtil(int board[N][N], int col)

{

if (col >= N)

return true;

count++;

for (int i = 0; i < N; i++)

{count++;

if (isSafe(board,i,col))

{

board[i][col] = 1;

count++;

if (solveNQUtil(board, col + 1))

return true;

count++;

board[i][col] = 0;

count++;

}count++;

}count++;

return false;

}

bool solveNQ()

{

int board[N][N] = { { 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 } };

count++;

if (solveNQUtil(board, 0) == false)

{

printf("Solution does not exist");

return false;

count++;

}

printSolution(board);

printf("count=%d",count);

return true;

}

int main()

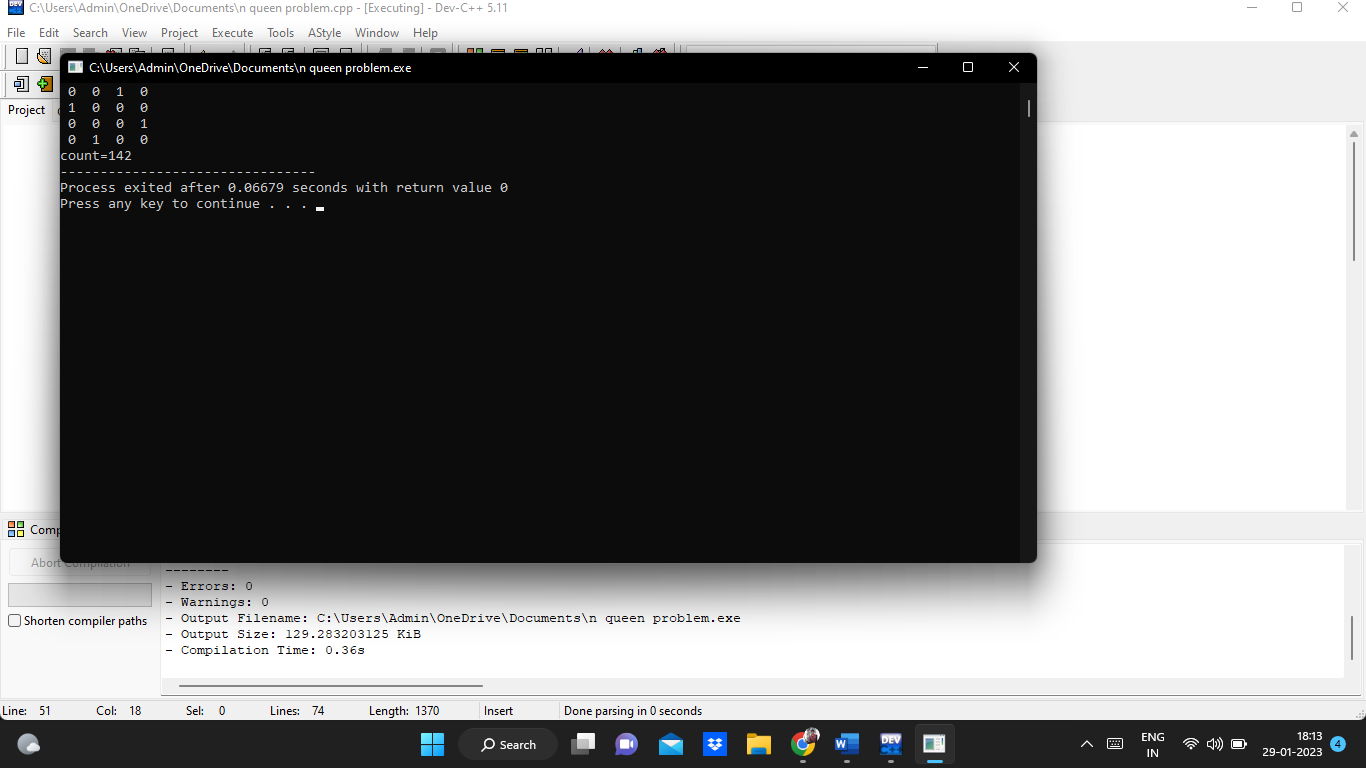
{

solveNQ();

return 0;

}

Output:



2. Write a program to find all pair shortest path using Floyds technique and to estimate time complexity.

Program:

#include <stdio.h>

#define nV 4

#define INF 999

void printMatrix(int matrix[][nV]);

void floydWarshall(int graph[][nV])

{

int matrix[nV][nV], i, j, k;

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

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

matrix[i][j] = graph[i][j];

for (k = 0; k < nV; k++) {

for (i = 0; i < nV; i++) {

for (j = 0; j < nV; j++) {

if (matrix[i][k] + matrix[k][j] < matrix[i][j])

matrix[i][j] = matrix[i][k] + matrix[k][j];

}

}

}

printMatrix(matrix);

}

void printMatrix(int matrix[][nV])

{

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

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

if (matrix[i][j] == INF)

printf("%4s", "INF");

else

printf("%4d", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int graph[nV][nV] = {{0, 3, INF, 5},

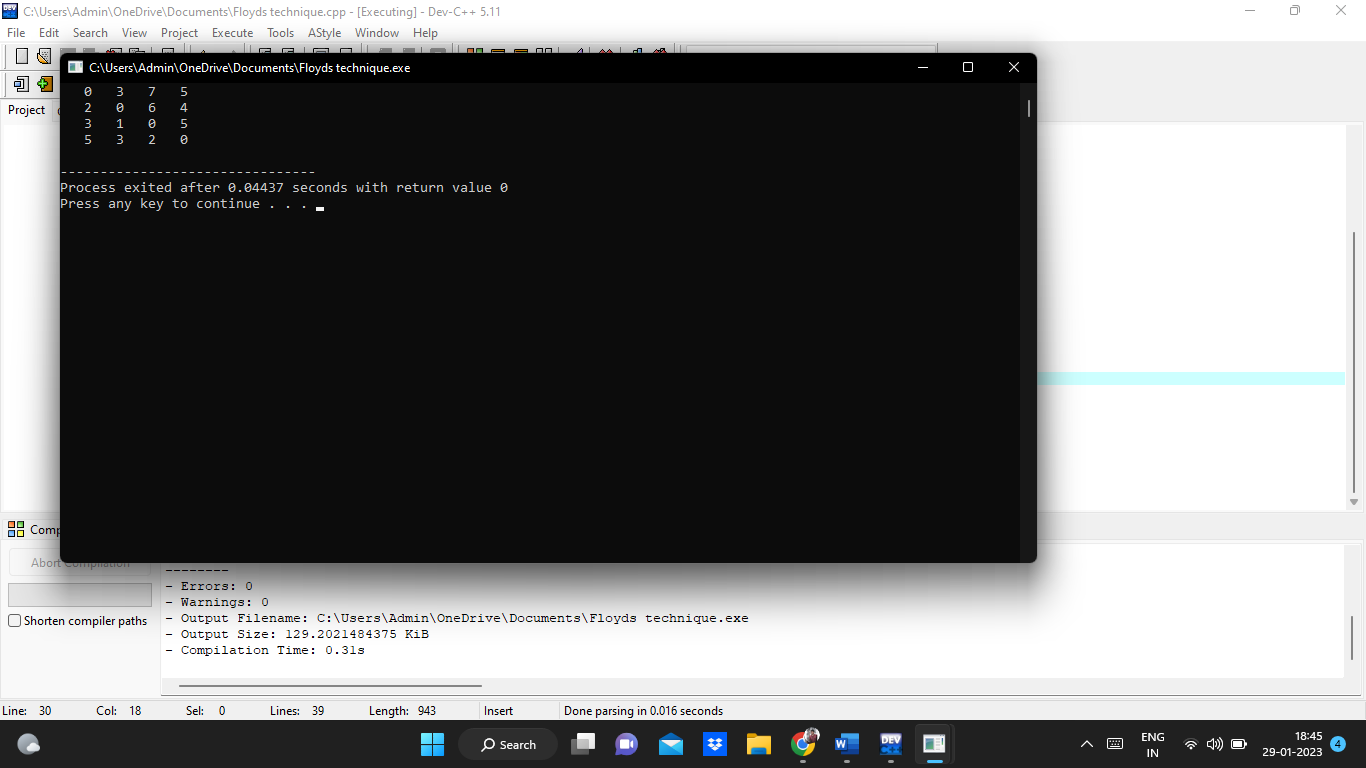
{2, 0, INF, 4},

{INF, 1, 0, INF},

{INF, INF, 2, 0}};

floydWarshall(graph);

}



3. #include <stdio.h>

int count=0;

int max(int a, int b)

{

count++;

return (a > b) ? a : b;

}

int knapSack(int W, int wt[], int val[], int n)

{

count++;

if (n == 0 || W == 0)

return 0;

count++;

if (wt[n - 1] > W)

return knapSack(W, wt, val, n - 1);

else

return max(

val[n - 1]

+ knapSack(W - wt[n - 1], wt, val, n - 1),

knapSack(W, wt, val, n - 1));

count++;

}

int main()

{

int val[] = { 60, 100, 120 };

count++;

int wt[] = { 10, 20, 30 };

count++;

int W = 50;

count++;

int n = sizeof(val) / sizeof(val[0]);

count++;

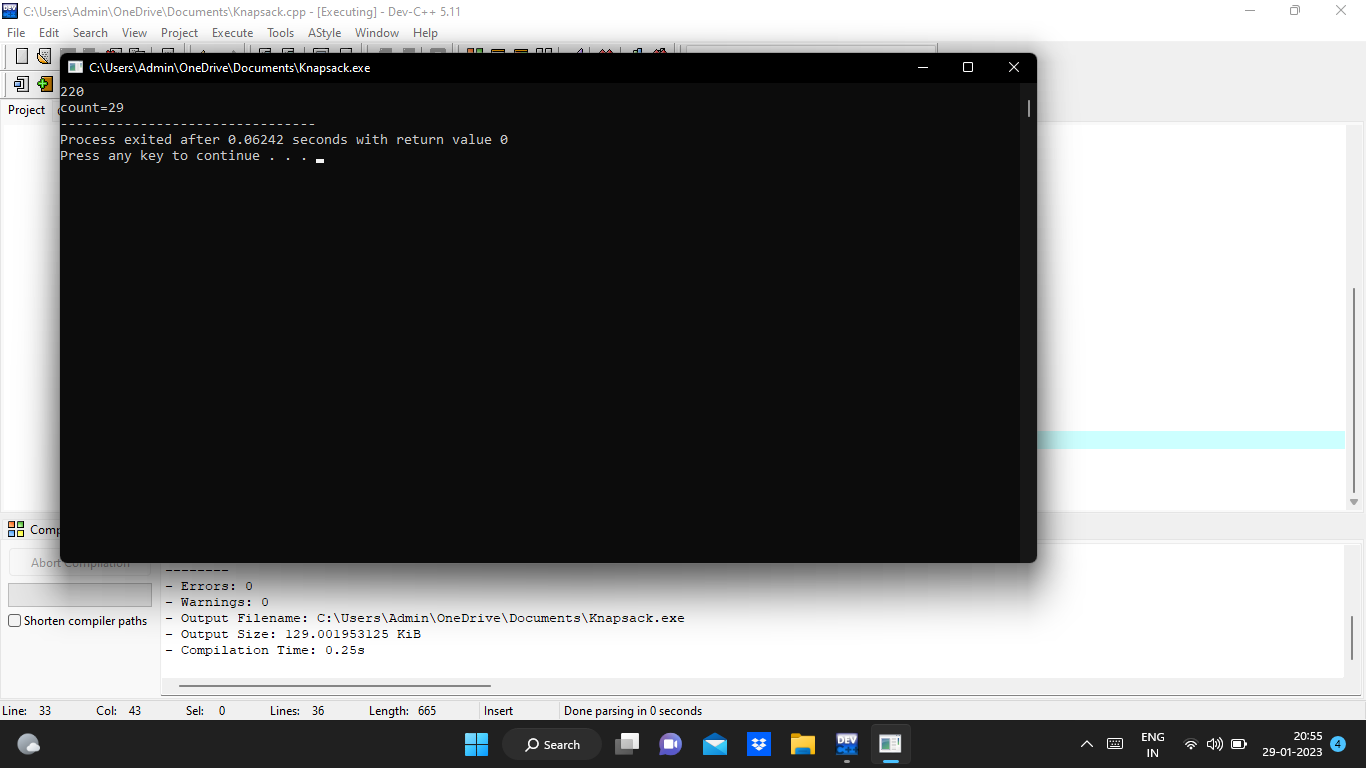
printf("%d", knapSack(W, wt, val, n));

printf("\ncount=%d",count);

return 0;

}

Output:



4.Travelling salesman problem.

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");

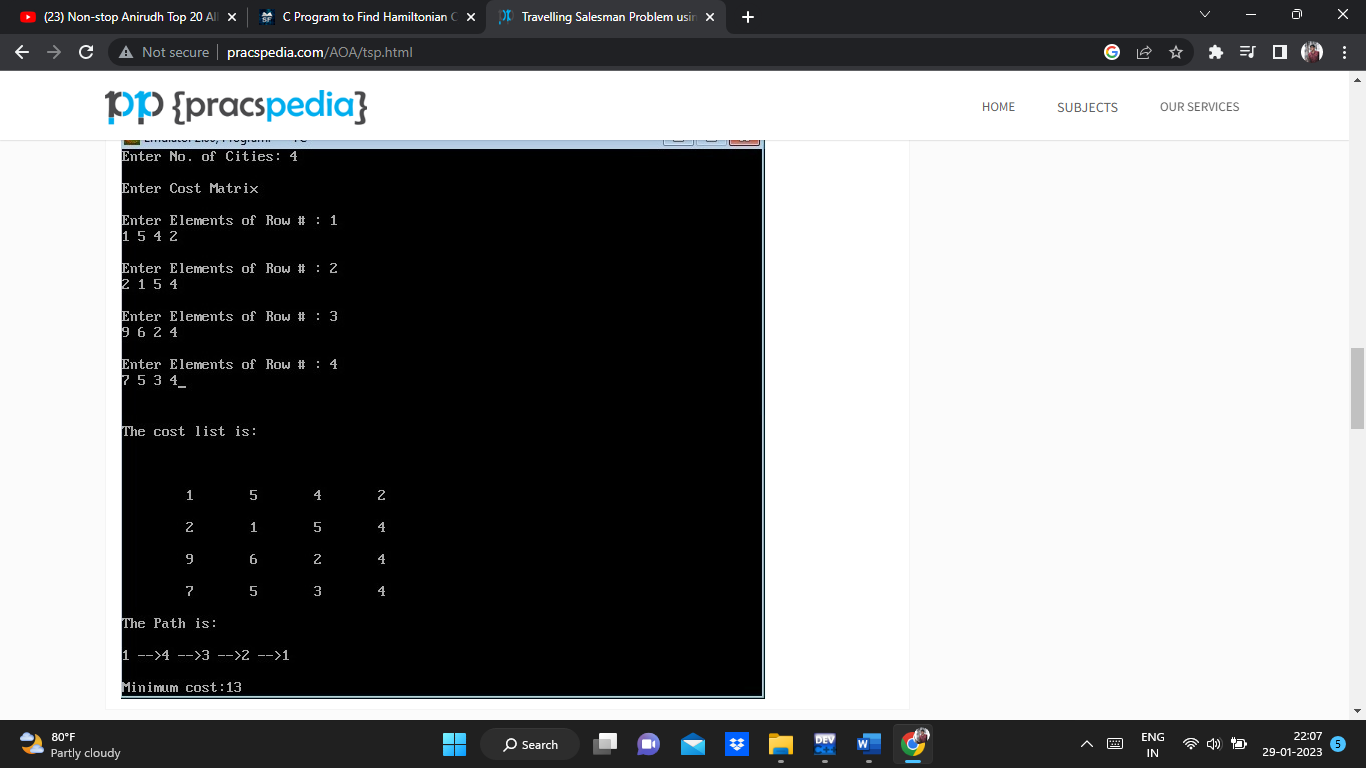
mincost(0);

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

return 0;

}

Output:



5. Write a program to find a minimum spanning tree using prims technique for the

given graph.

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];

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;

}

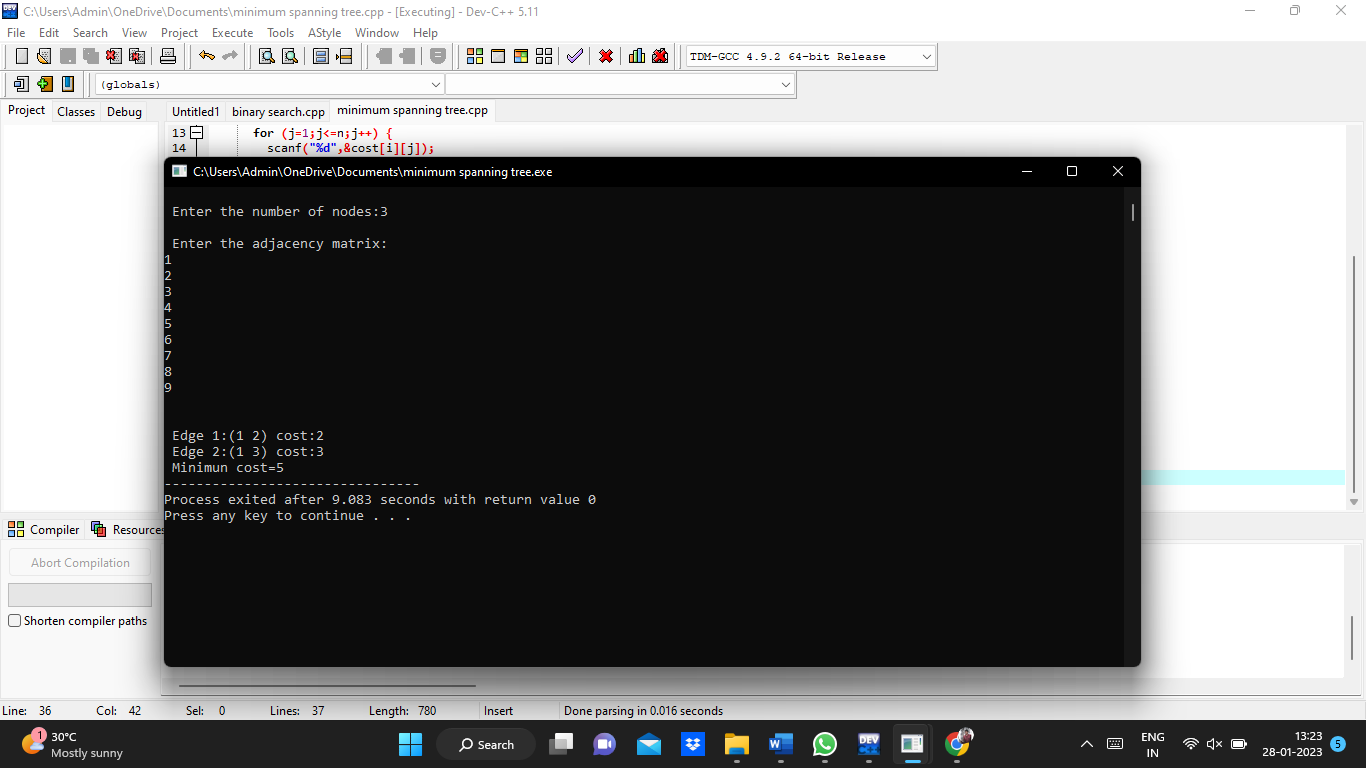
cost[a][b]=cost[b][a]=999;

}

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

}

Output :



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

Program:

#include<stdio.h>

#include<stdlib.h>

#define V 5

int count=0;

void printSolution(int path[]);

int isSafe(int v, int graph[V][V], int path[], int pos)

{count++;

if (graph[path[pos - 1]][v] == 0)

return 0;

count++;

int i;

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

if (path[i] == v)

return 0;

return 1;

}

int hamCycleUtil(int graph[V][V], int path[], int pos) {

if (pos == V) {

if (graph[path[pos - 1]][path[0]] == 1)

return 1;

else

return 0;

}

int v;

for (v = 1; v < V; v++) {

if (isSafe(v, graph, path, pos)) {

path[pos] = v;

count++;

if (hamCycleUtil(graph, path, pos + 1) == 1)

return 1;

path[pos] = -1;

count++;

}

}

return 0;

}

int hamCycle(int graph[V][V]) {

int \*path = new int[V];

count++;

int i;

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

path[i] = -1;

count++;

path[0] = 0;

count++;

if (hamCycleUtil(graph, path, 1) == 0) {

printf("\nSolution does not exist");

return 0;

}

printSolution(path);

return 1;

}

void printSolution(int path[]) {

printf("Solution Exists:"

" Following is one Hamiltonian Cycle \n");

int i;

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

printf(" %d ", path[i]);

printf(" %d ", path[0]);

printf("\n");

}

int main() {

int graph1[V][V] = { { 0, 1, 0, 1, 0 }, { 1, 0, 1, 1, 1 },

{ 0, 1, 0, 0, 1 }, { 1, 1, 0, 0, 1 }, { 0, 1, 1, 1, 0 }, };

count++;

hamCycle(graph1);

count++;

int graph2[V][V] = { { 0, 1, 0, 1, 0 }, { 1, 0, 1, 1, 1 },

{ 0, 1, 0, 0, 1 }, { 1, 1, 0, 0, 0 }, { 0, 1, 1, 0, 0 }, };

count++;

hamCycle(graph2);

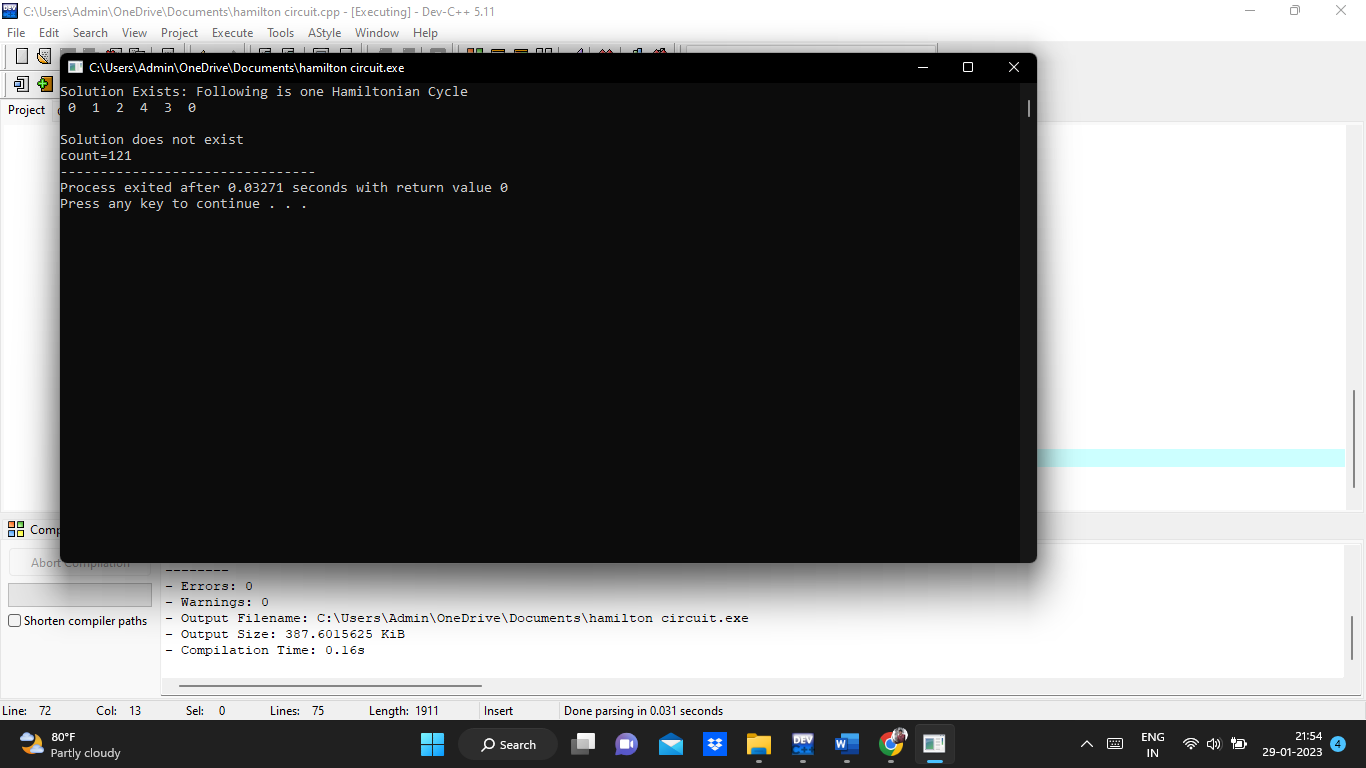
count++;

printf("\ncount=%d",count);

return 0;

}

Output:



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

Program :

#include<stdio.h>

int count=0;

int max, min;

int a[100];

void maxmin(int i, int j)

{

count++;

int max1, min1, mid;

if(i==j)

{count++;

max = min = a[i];

count++;

}

else

{count++;

if(i == j-1)

{count++;

if(a[i] <a[j])

{count++;

max = a[j];

count++;

min = a[i];

count++;

}

else

{count++;

max = a[i];

count++;

min = a[j];

count++;

}

}

else

{count++;

mid = (i+j)/2;

count++;

maxmin(i, mid);

count++;

max1 = max; min1 = min;

count++;

maxmin(mid+1, j);

count++;

if(max <max1)

max = max1;

count++;

if(min > min1)

min = min1;

count++;

}count++;

}count++;

}

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++)

{

count++;

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

}count++;

max = a[0];

count++;

min = a[0];

count++;

maxmin(1, num);

count++;

printf ("Minimum element in an array : %d\n", min);

printf ("Maximum element in an array : %d\n", max);

printf("count=%d",count);

return 0;

}

Output :

