#include <stdio.h>

void binary\_search(int [], int, int, int);

void bubble\_sort(int [], int);

int main()

{

int element, size, i;

int list[25];

printf("Enter size of a list: ");

scanf("%d", &size);

printf("Enter elements\n");

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

{

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

}

bubble\_sort(list, size);

printf("\n");

printf("Enter element to search\n");

scanf("%d", &element);

binary\_search(list, 0, size, element);

}

void bubble\_sort(int list[], int size)

{

int temp, i, j;

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

{

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

{

if (list[i] > list[j])

{

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

}

}

void binary\_search(int list[], int lo, int hi, int element)

{

int mid;

if (lo > hi)

{

printf("element not found\n");

return;

}

mid = (lo + hi) / 2;

if (list[mid] == element)

{

printf("element found\n");

}

else if (list[mid] > element)

{

binary\_search(list, lo, mid - 1, element);

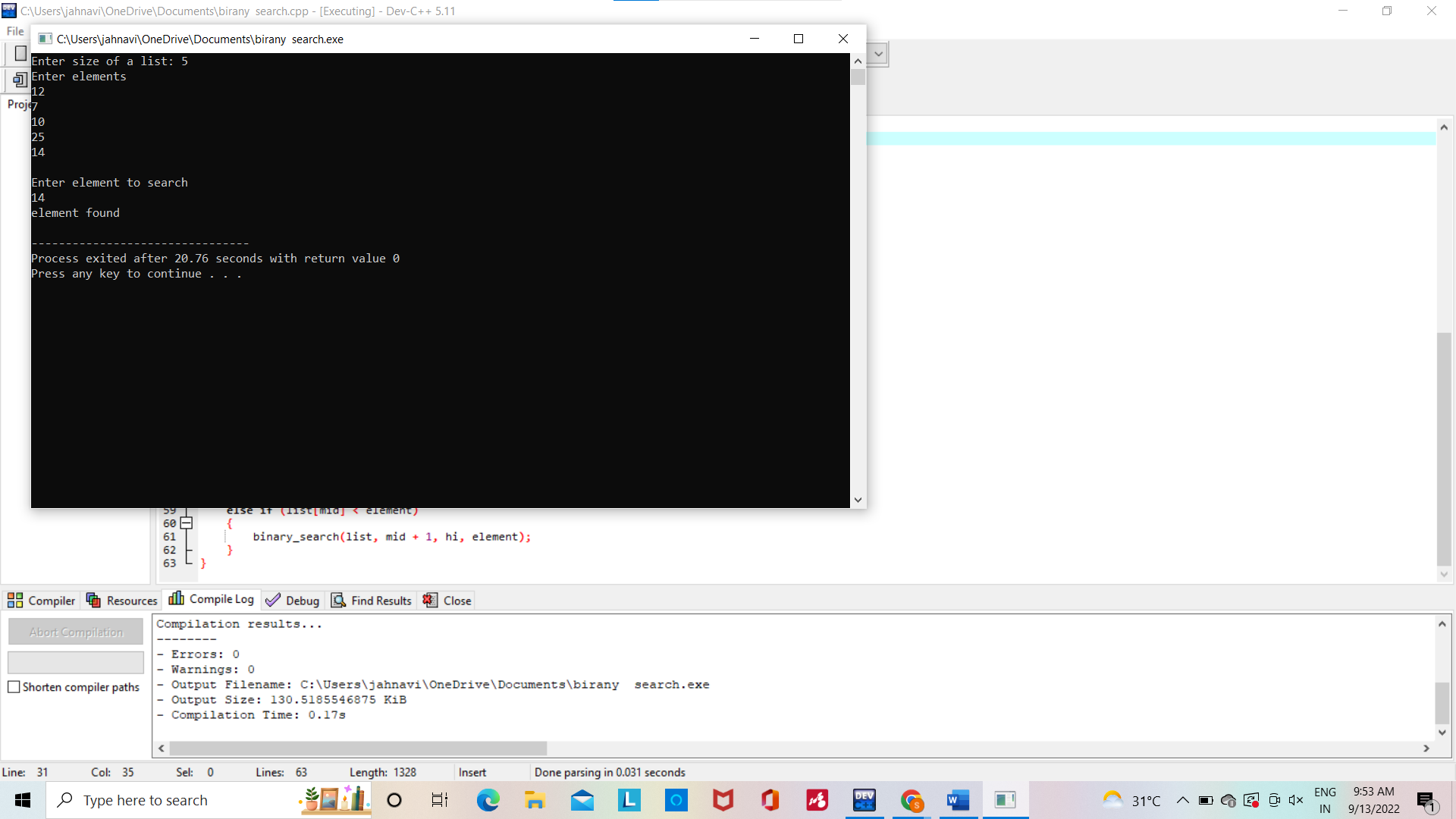
}

else if (list[mid] < element)

{

binary\_search(list, mid + 1, hi, element);

}



MERGE SORT:

#include <stdio.h>

void merge(int arr[], int start, int mid, int end)

{

int i,j,k;

int len1 = mid - start + 1;

int len2 = end - mid;

int leftArr[len1], rightArr[len2];

for ( i = 0; i < len1; i++)

leftArr[i] = arr[start + i];

for (j = 0; j < len2; j++)

rightArr[j] = arr[mid + 1 + j];

i = 0;

j = 0;

k = start;

while (i < len1 && j < len2)

{

if (leftArr[i] <= rightArr[j])

{

arr[k] = leftArr[i];

i++;

}

else

{

arr[k] = rightArr[j];

j++;

}

k++;

}

while (i < len1) {

arr[k] = leftArr[i];

i++;

k++;

}

while (j < len2) {

arr[k] = rightArr[j];

j++;

k++;

}

}

void mergeSort(int arr[], int start, int end) {

if (start < end) {

int mid = start + (end - start) / 2;

mergeSort(arr, start, mid);

mergeSort(arr, mid + 1, end);

merge(arr, start, mid, end);

}

}

void display(int arr[], int size)

{

int i;

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

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

printf("\n");

}

int main() {

int arr[] = {6, 5, 12, 10, 9, 1 , 4};

int size = sizeof(arr) / sizeof(arr[0]);

printf("Original array\n");

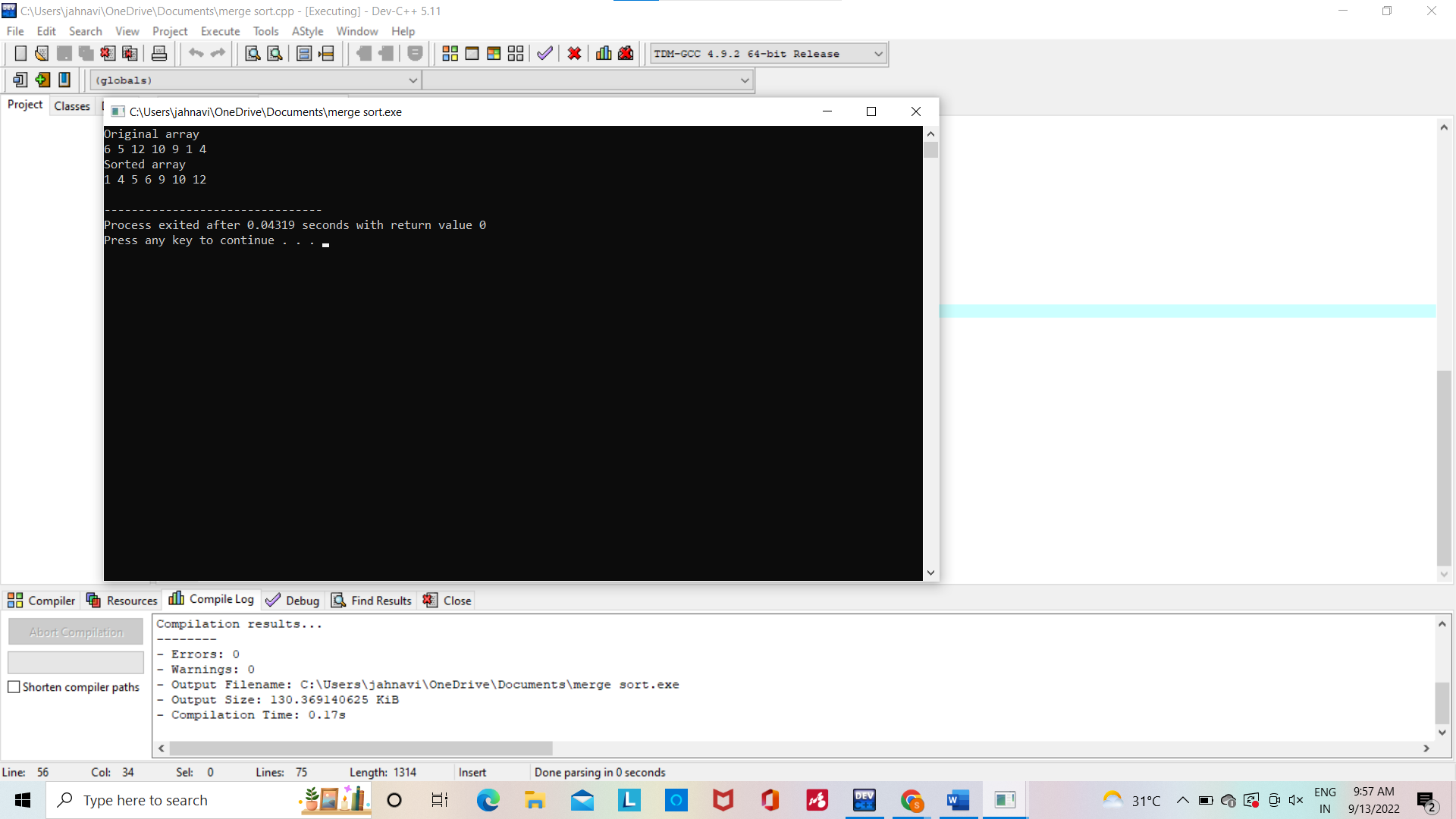
display(arr, size);

mergeSort(arr, 0, size - 1);

printf("Sorted array\n");

display(arr, size);

}



STRASSENS MATRIX MULTIPLICATION:

#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");

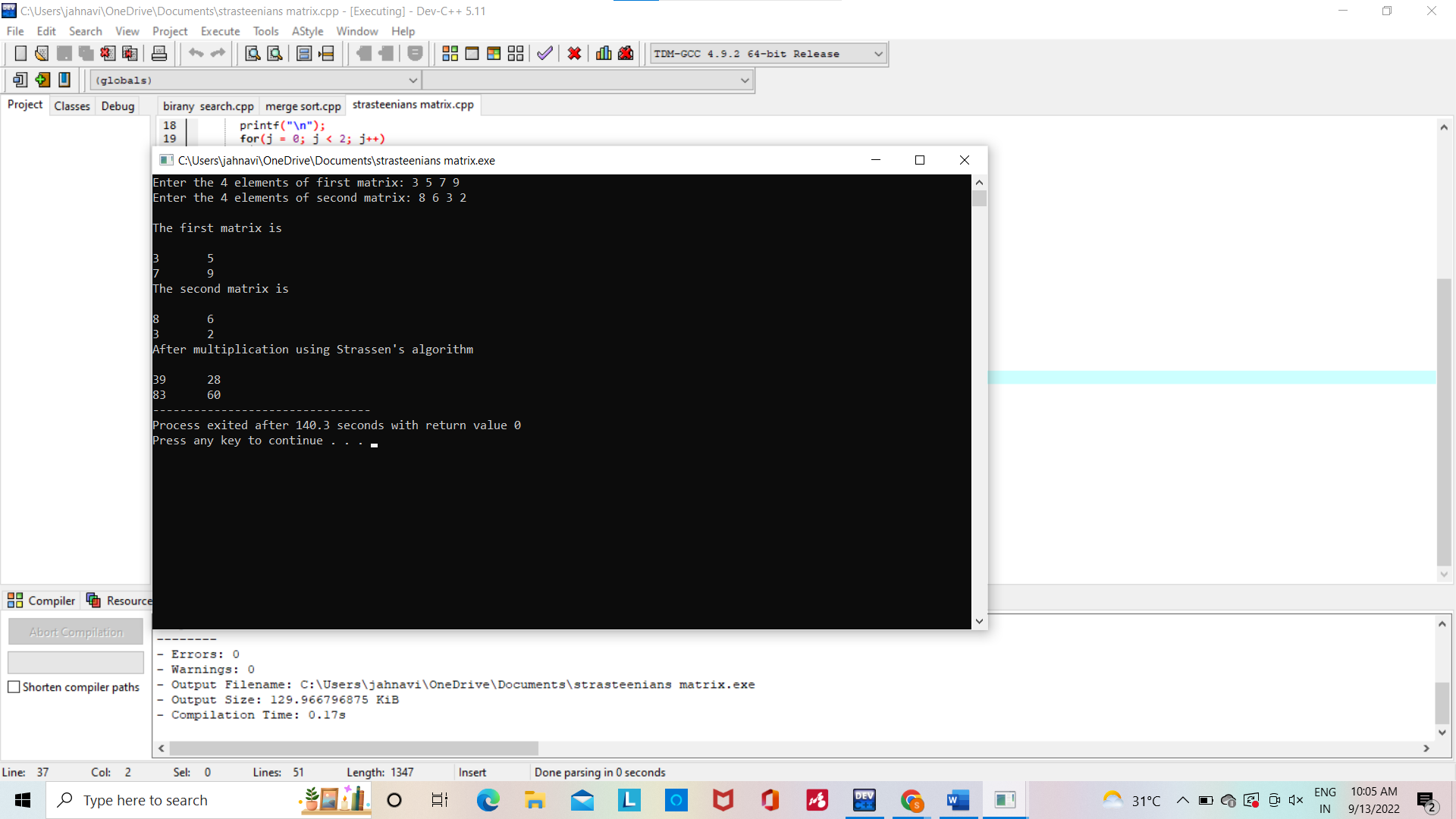
for(j = 0;j < 2; j++)

printf("%d\t", c[i][j]);

}

return 0;

}



PRIMS ALGORITHM:

#include<stdio.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;

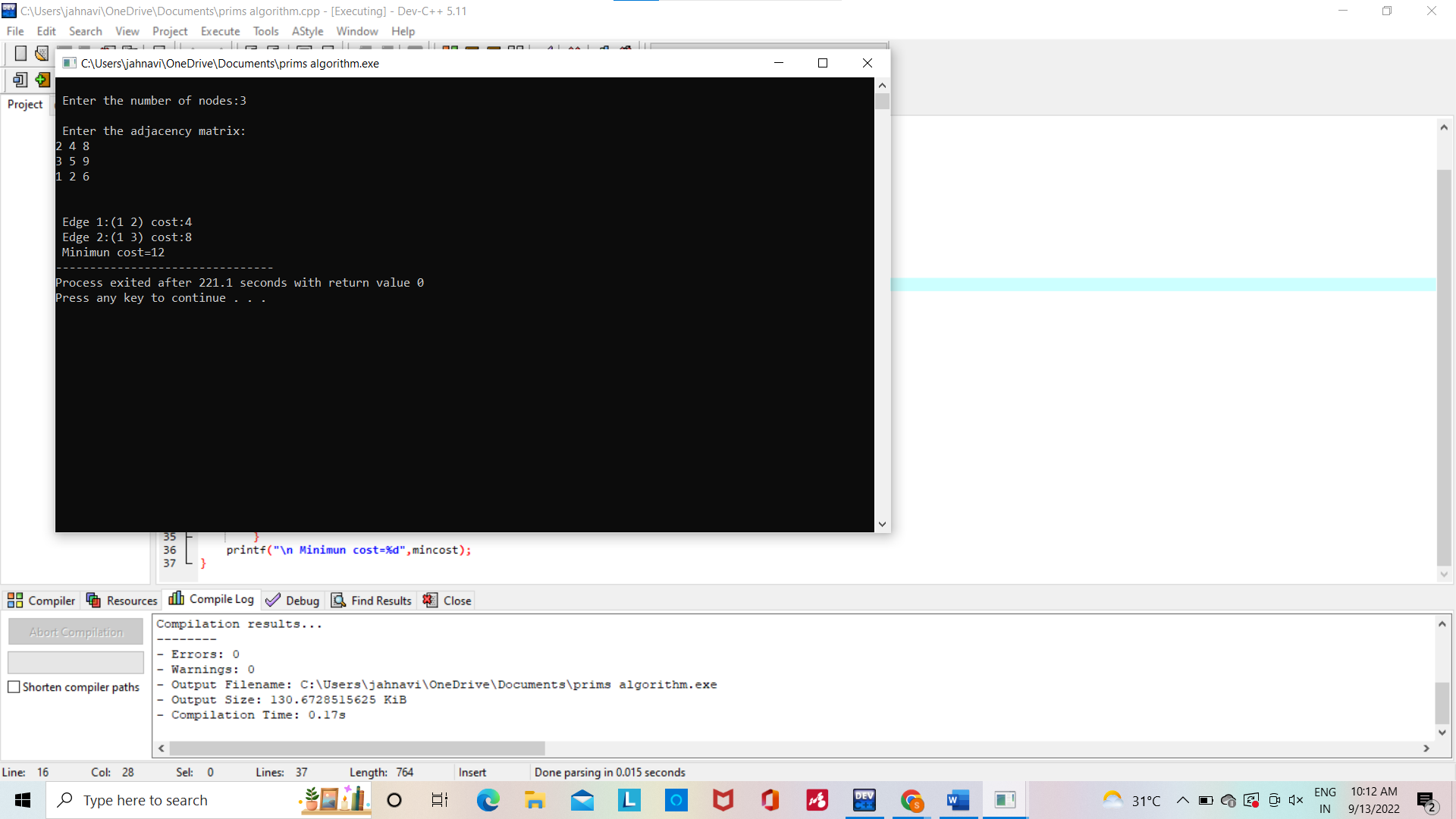
}

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

}

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

}



KNAPSACK USING GREEDY:

#include<stdio.h>

int max(int a, int b)

{

return (a > b)? a : b;

}

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

{

int i, w;

int K[n+1][W+1];

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

{

for (w = 0; w <= W; w++)

{

if (i==0 || w==0)

K[i][w] = 0;

else if (wt[i-1] <= w)

K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

else

K[i][w] = K[i-1][w];

}

}

return K[n][W];

}

int main()

{

int i, n, val[20], wt[20], W;

printf("Enter number of items:");

scanf("%d", &n);

printf("Enter value and weight of items:\n");

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

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

}

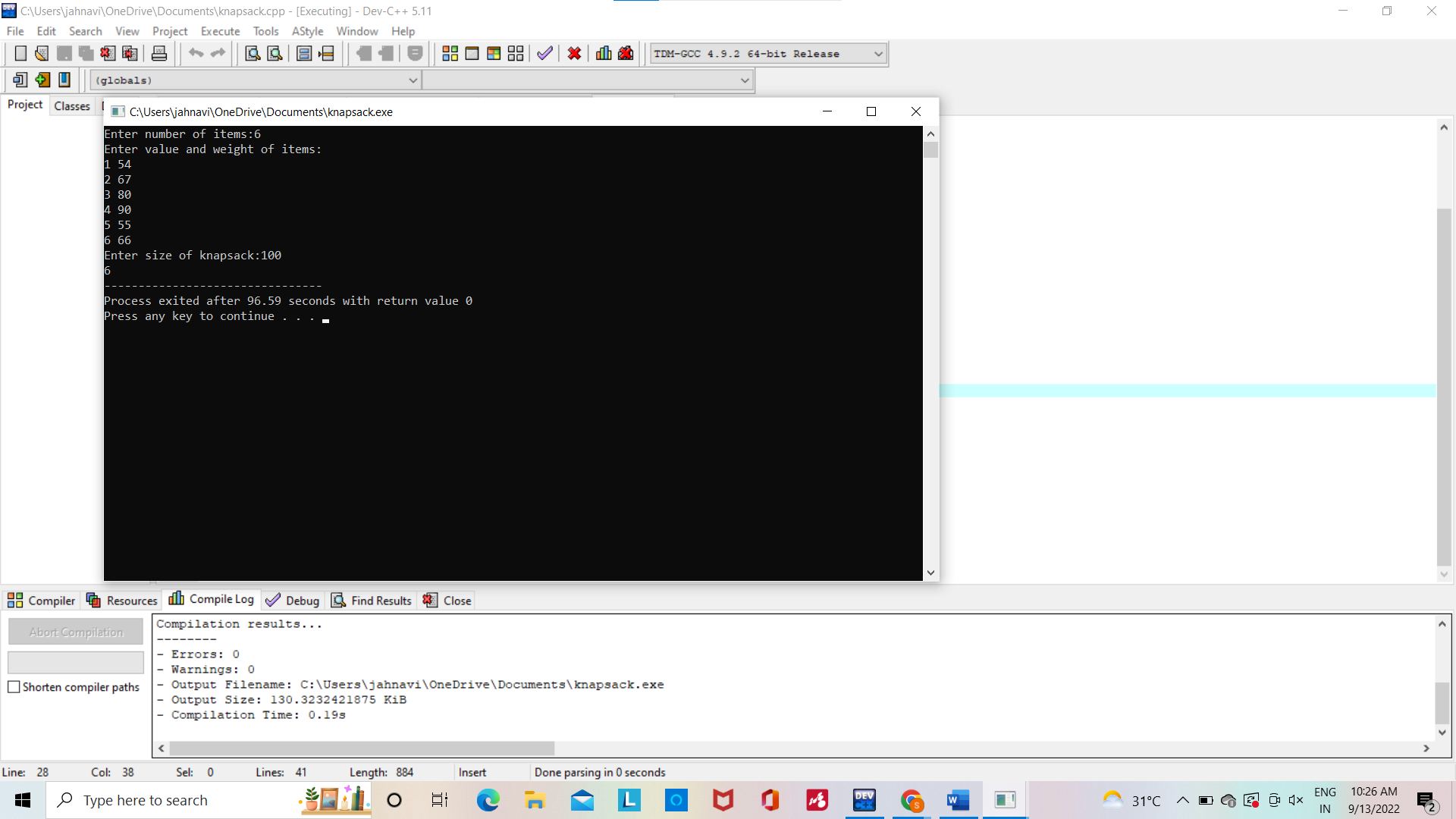
printf("Enter size of knapsack:");

scanf("%d", &W);

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

return 0;

}



GCD OF TWO NUMBERS:

#include <stdio.h>

int main()

{

int n1, n2, i, gcd;

printf("Enter two integers: ");

scanf("%d %d", &n1, &n2);

for(i=1; i <= n1 && i <= n2; ++i)

{

// Checks if i is factor of both integers

if(n1%i==0 && n2%i==0)

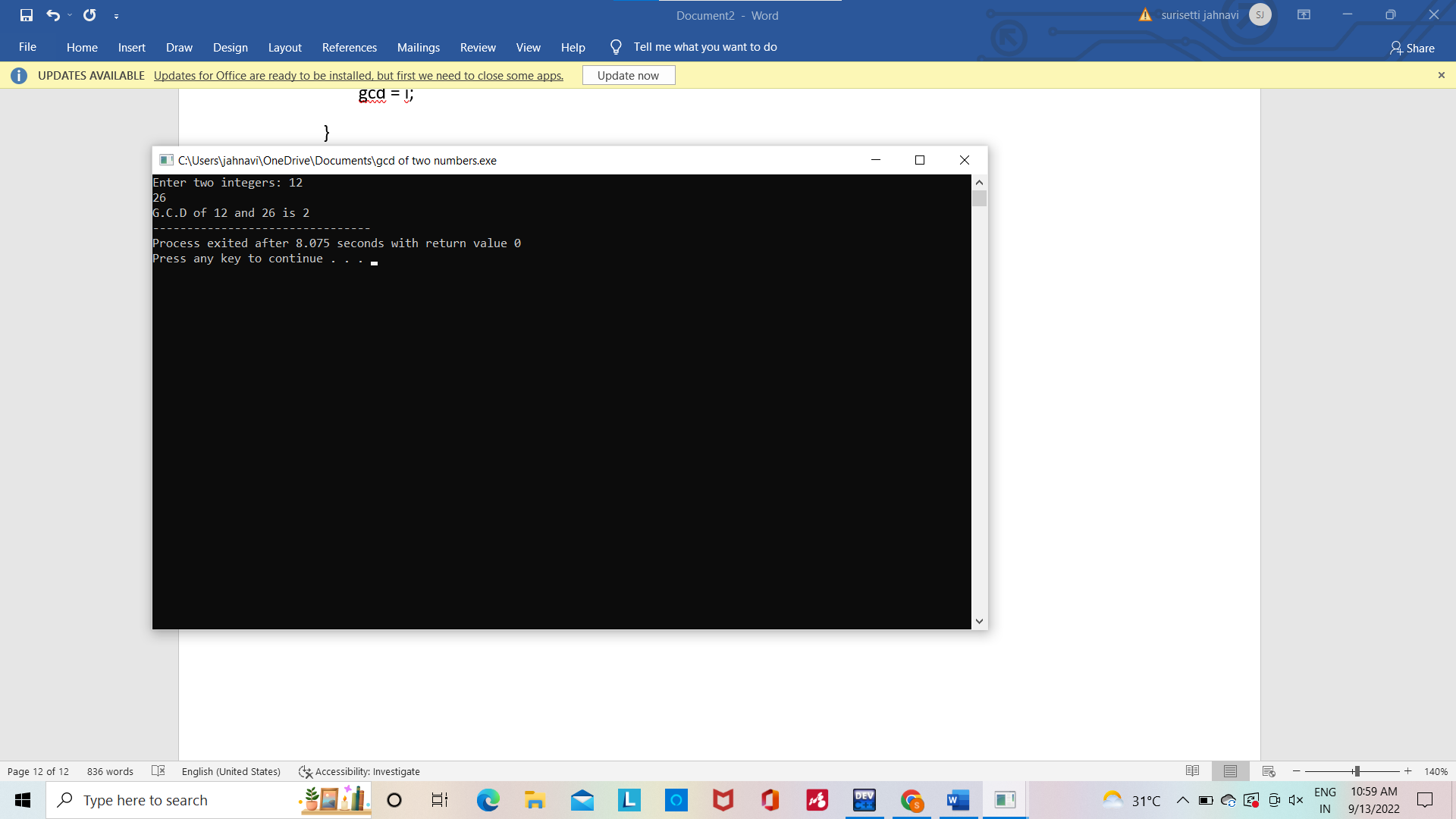
gcd = i;

}

printf("G.C.D of %d and %d is %d", n1, n2, gcd);

return 0;

}



FIBANOCCI SERIES:

#include<stdio.h>

void printFibonacci(int n){

static int n1=0,n2=1,n3;

if(n>0){

n3 = n1 + n2;

n1 = n2;

n2 = n3;

printf("%d ",n3);

printFibonacci(n-1);

}

}

int main(){

int n;

printf("Enter the number of elements: ");

scanf("%d",&n);

printf("Fibonacci Series: ");

printf("%d %d ",0,1);

printFibonacci(n-2);

return 0;

}