**DISCRETE STRUCTURE AND LOGIC LAB (KCS-353) FOR**

**ACADEMIC YEAR 2021-22**



# BUNDELKHAND INSTITUTE OF ENGINEERING AND TECHNOLOGY, JHANSI

**(BIET JHANSI)-284128**

**S ubmitted to:- S ubmitted by:-**

# Er. Sharad Kumar Ankush Kumar Kushwaha

**Roll No. 2004310015**

**INDEX**

**Name:** Pranshu Mishra **Roll No.:** 1904310041

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Date** | **Aim/Objective** | **T.Sign** |
| 1 |  | Write a program to create two sets and perform the Union operation on sets. |  |
| 2 |  | Write a program to create two sets and perform the intersection operation on sets. |  |
| 3 |  | Write a program to create two sets and perform the Difference operation on sets. |  |
| 4 |  | Write a program to create two sets and perform the Symmetric Difference operation. |  |
| 5 |  | Write a program to perform the Power Set operation. |  |
| 6 |  | Write a program to display the Truth Table for AND, OR, NOT. |  |
| 7 |  | Write a program to find cartesian product of two sets. |  |
| 8 |  | Write a program for minimum cost spanning tree. |  |
| 9 |  | Write a program for finding the shortest path in a graph. |  |

# PRACTICAL NO. 1

**Aim :- To write a program in C language to create two sets and perform the Union operation on sets.**

**Program:-**

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

int a[10],b[10],m,n,i,j; int c[20],k=0,flag=0; int ch;

printf("Enter the number of elements in first set:\n"); scanf("%d",&m);

printf("Enter the elements:\n"); for(i=0;i<m;i++){

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

}

printf("\nElement of First set:\n");

for(i=0;i<m;i++){ printf("%d\t",a[i]);

}

printf("\nEnter the number of elements in second set:\n"); scanf("%d",&n);

printf("Enter the elements:\n"); for(i=0;i<n;i++) {

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

}

printf("\nElement of Second set:\n"); for(i=0;i<n;i++) {

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

}

for(i=0;i<m;i++){ c[k]=a[i];

k++;

}

for(i=0;i<n;i++){ flag=0; for(j=0;j<m;j++)

{

if(b[i]==c[j])

{

flag=1; break;

}

}

if(flag==0)

{

c[k]=b[i]; k++;

}

}

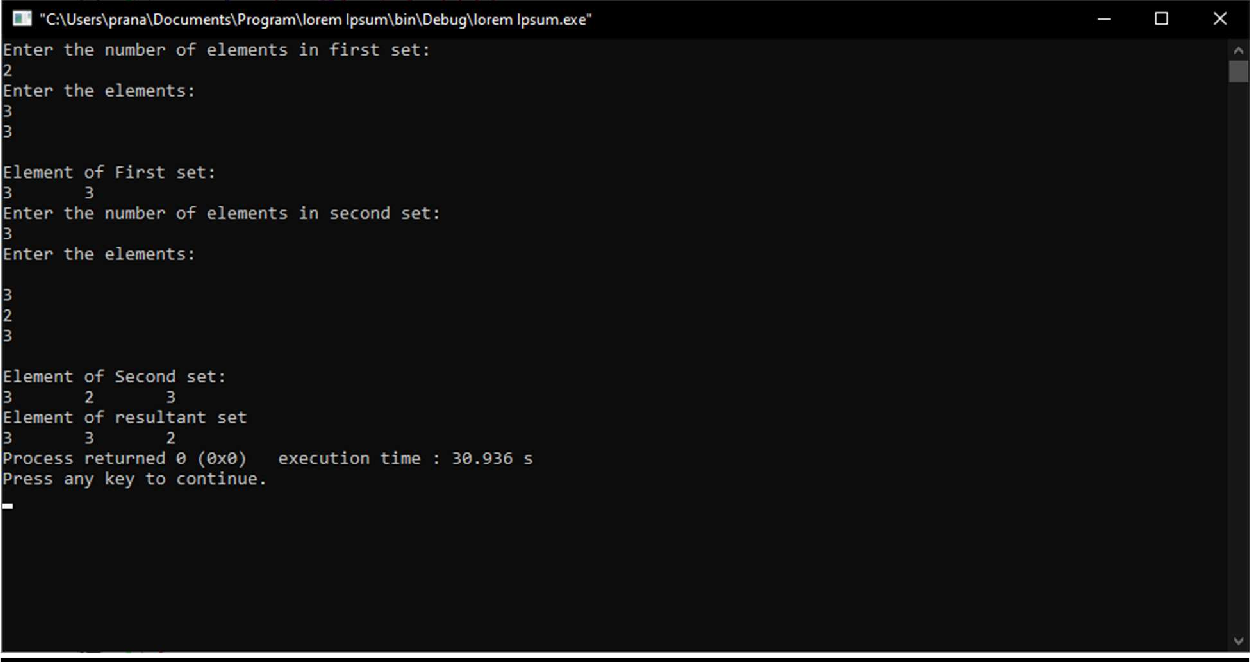
printf("\nElement of resultant set\n"); for(i=0;i<k;i++){

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

}

}

# OUTPUT



**PRACTICAL NO. 2**

**Aim :- To write a program in C language to create two sets and perform the intersection operation on sets.**

**Program :-**

#include <stdio.h> int main()

{

int a[10], b[10], flag = 0, n1, n2, i, j; printf("Enter array1 size : "); scanf("%d",&n1);

printf("\nEnter array2 size : "); scanf("%d",&n2);

printf("\nEnter array1 element : "); for(i = 0;i < n1;i++) scanf("%d",&a[i]);

printf("\nEnter array2 element : "); for(i = 0;i < n2;i++) scanf("%d",&b[i]); printf("Intersection: ");

for(i = 0;i < n1;i++)

{

for(j = 0;j < n2;j++)

{

if(b[i] == a[j])

{

flag = 1;

}

}

if(flag == 1)

{

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

}

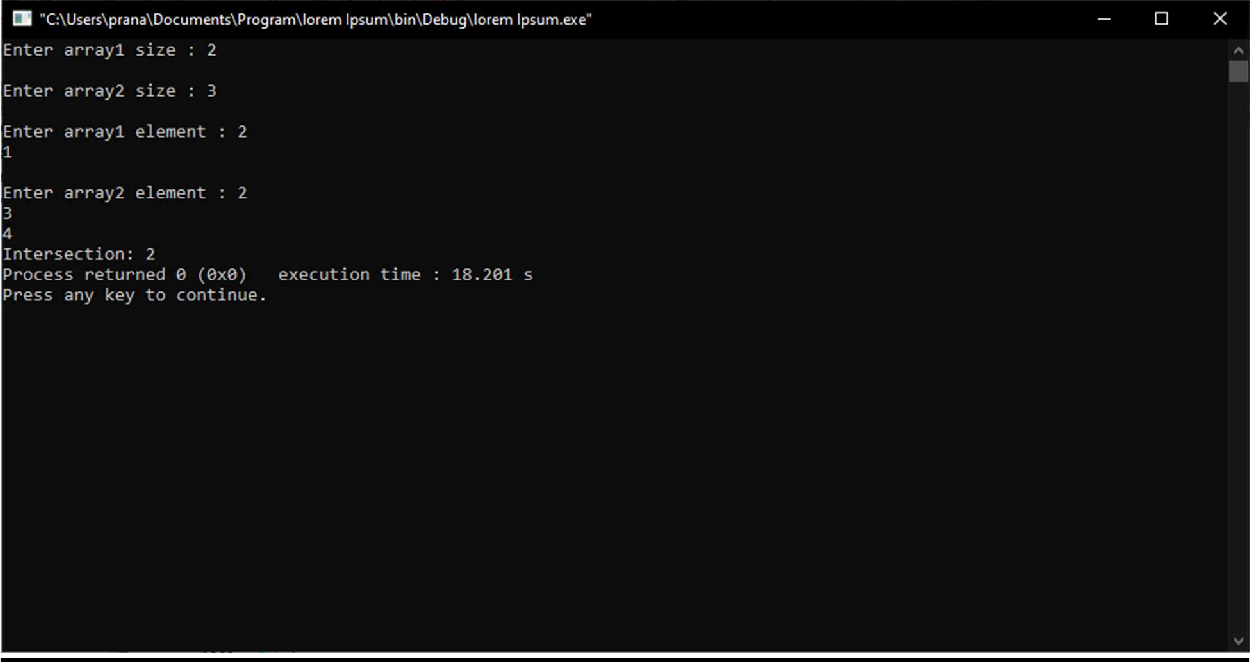
flag = 0;

}

return 0;

}

# OUTPUT



**PRACTICAL NO. 3**

**Aim :- Write a program to create two sets and perform the Difference operation on sets.**

**Program:-**

#include<stdio.h> #define max 100

int ifexists(int z[], int u, int v)

{

int i;

if (u==0) return 0; for (i=0; i<=u;i++)

if (z[i]==v) return (1); return (0);

}

void main()

{

int p[max], q[max], r[max]; int m,n;

int i,j,k;

printf("Enter length of first array:"); scanf("%d",&m);

printf("Enter %d elements of first array\n",m); for(i=0;i<m;i++ )

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

printf("\nEnter length of second array:"); scanf("%d",&n);

printf("Enter %d elements of second array\n",n); for(i=0;i<n;i++ )

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

k=0;

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

{

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

{

if (p[i]==q[j]) { break;

}

}

if( j== n)

{

if(!ifexists(r,k,p[i]))

{

r[k]=p[i]; k++;

}

}

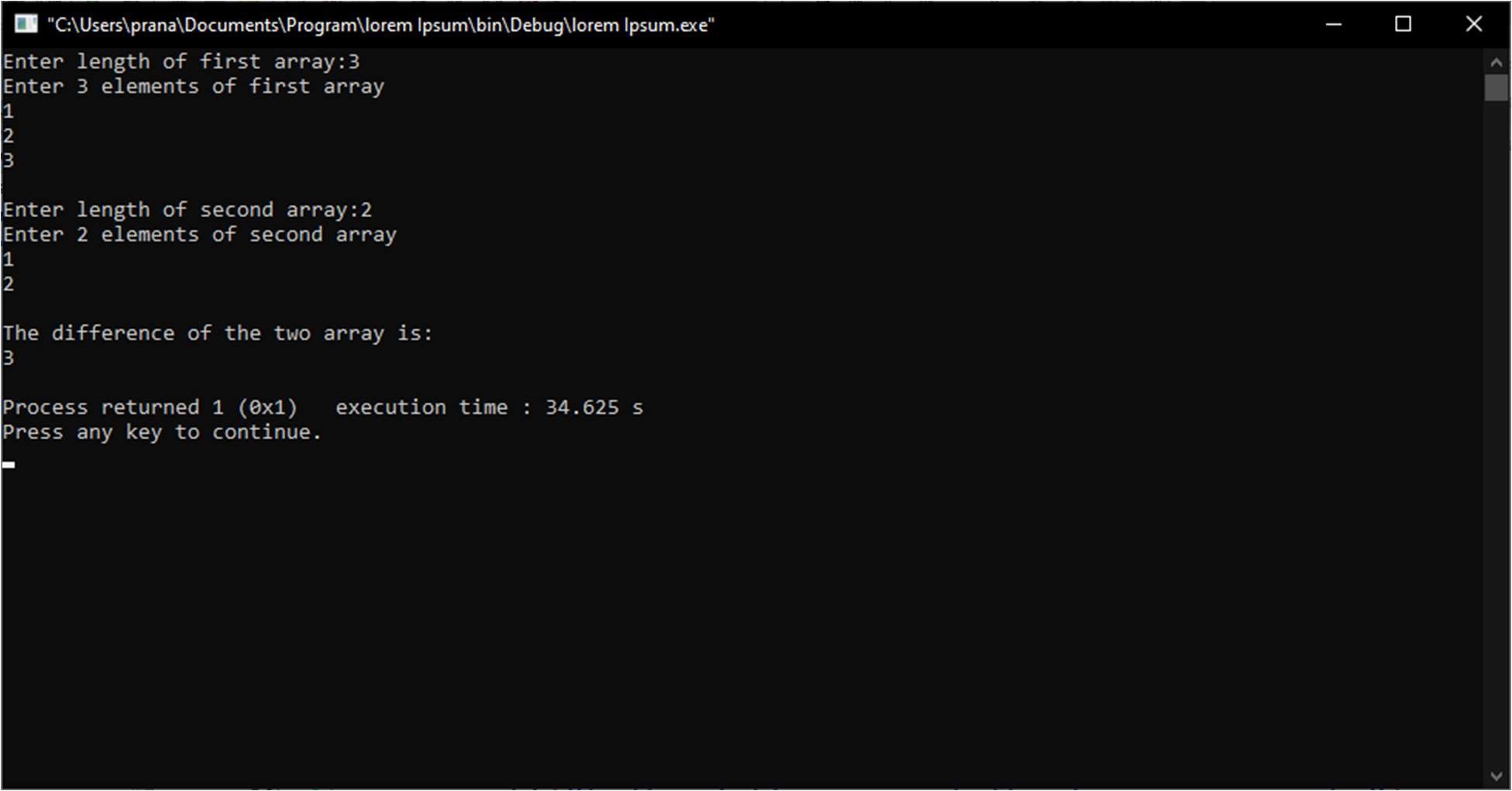
}

printf("\nThe difference of the two array is:\n"); for(i = 0;i<k;i++)

printf("%d\n",r[i]);

}

# OUTPUT



**PRACTICAL NO. 4**

**Aim:- Write a program to create two sets and perform the Symmetric Difference operation.**

**Program:-**

#include<stdio.h>

int a[10],b[10],c[10],d[10],i,j,k=0,n,m,flag=0; void symmetric\_diff()

{

k=0;

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

{

flag=0; for(j=0;j<m;j++)

{

if(a[i]==b[j])

{

flag=1; break;

}

}

if(flag==0)

{

d[k]=a[i]; k++;

}

}

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

flag=0; for(j=0;j<n;j++){

if(b[i]==a[j]){ flag=1;

break;

}

}

if(flag==0){ d[k]=b[i]; k++;

}

}

printf("\n(A-B)U(B-A) equals: "); for(i=0;i<k;i++){

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

}

}

int main()

{

printf("Enter the size of array A\n"); scanf("%d",&n);

printf("Enter the element of First array A\n"); for(i=0;i<n;i++)

{

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

}

printf("Enter the size of array B\n"); scanf("%d",&m);

printf("Enter the elements of array B\n"); for(j=0;j<m;j++)

{

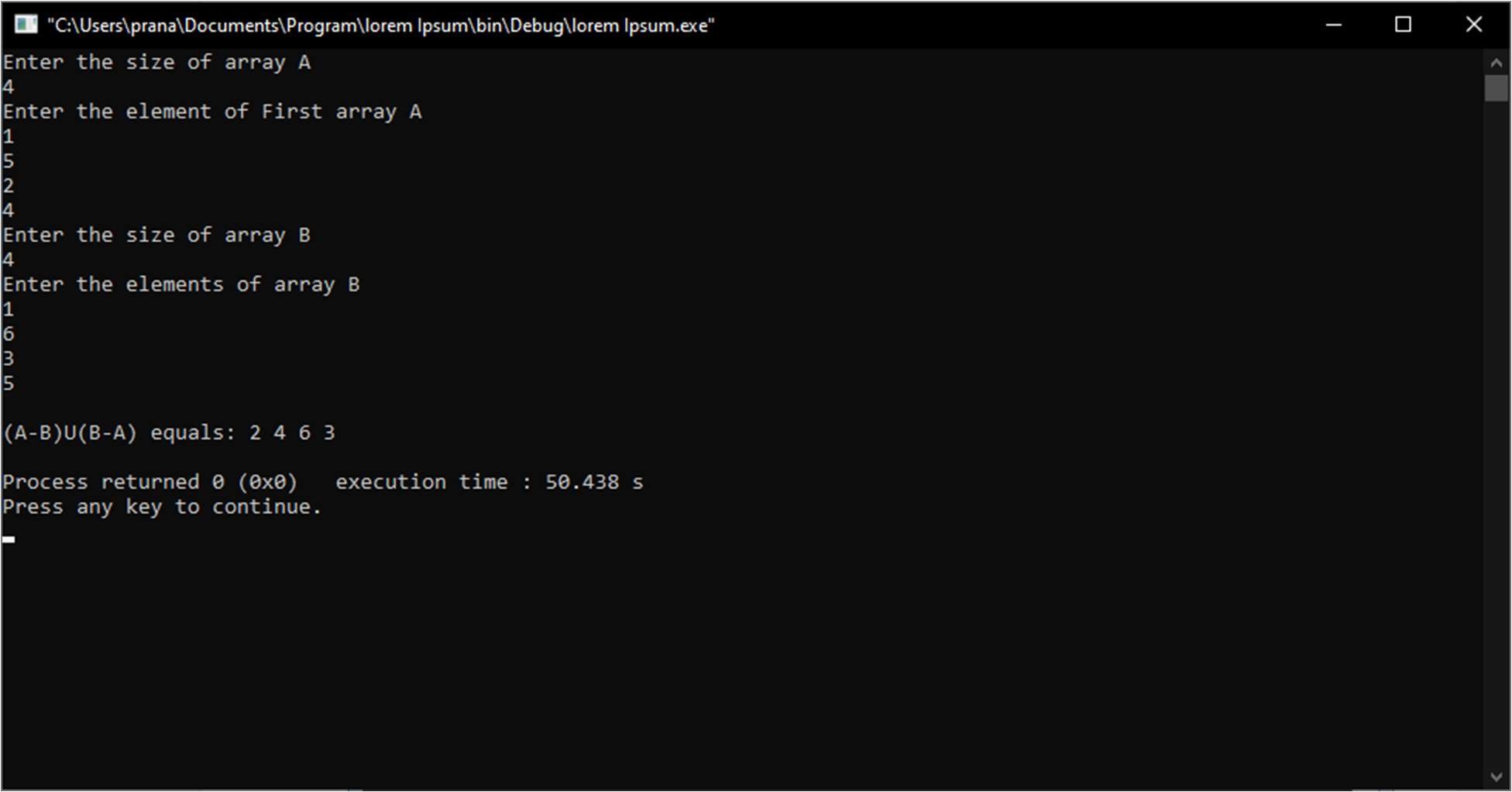
scanf("%d",&b[j]);

}

symmetric\_diff(); printf("\n"); return 0;

}

# OUTPUT



**PRACTICAL NO. 5**

**Aim :- Write a program to perform the Power Set operation.**

**Program:-**

#include <stdio.h> #include <math.h>

void printPowerSet(int \*set, int set\_size)

{

unsigned int pow\_set\_size = pow(2, set\_size); int counter, j;

for(counter = 0; counter < pow\_set\_size; counter++)

{

printf("{ ");

for(j = 0; j < set\_size; j++)

{

if(counter & (1<<j)) printf("%d ", set[j]);

}

printf("}");

printf("\n");

}

}

int main()

{

int n, i, \*set;

printf("Enter the size of set:\n"); scanf("%d",&n);

set = (int\*)malloc(n\*sizeof(int)); printf("Enter the elements in the set:\n"); for(i=0;i<n;i++){

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

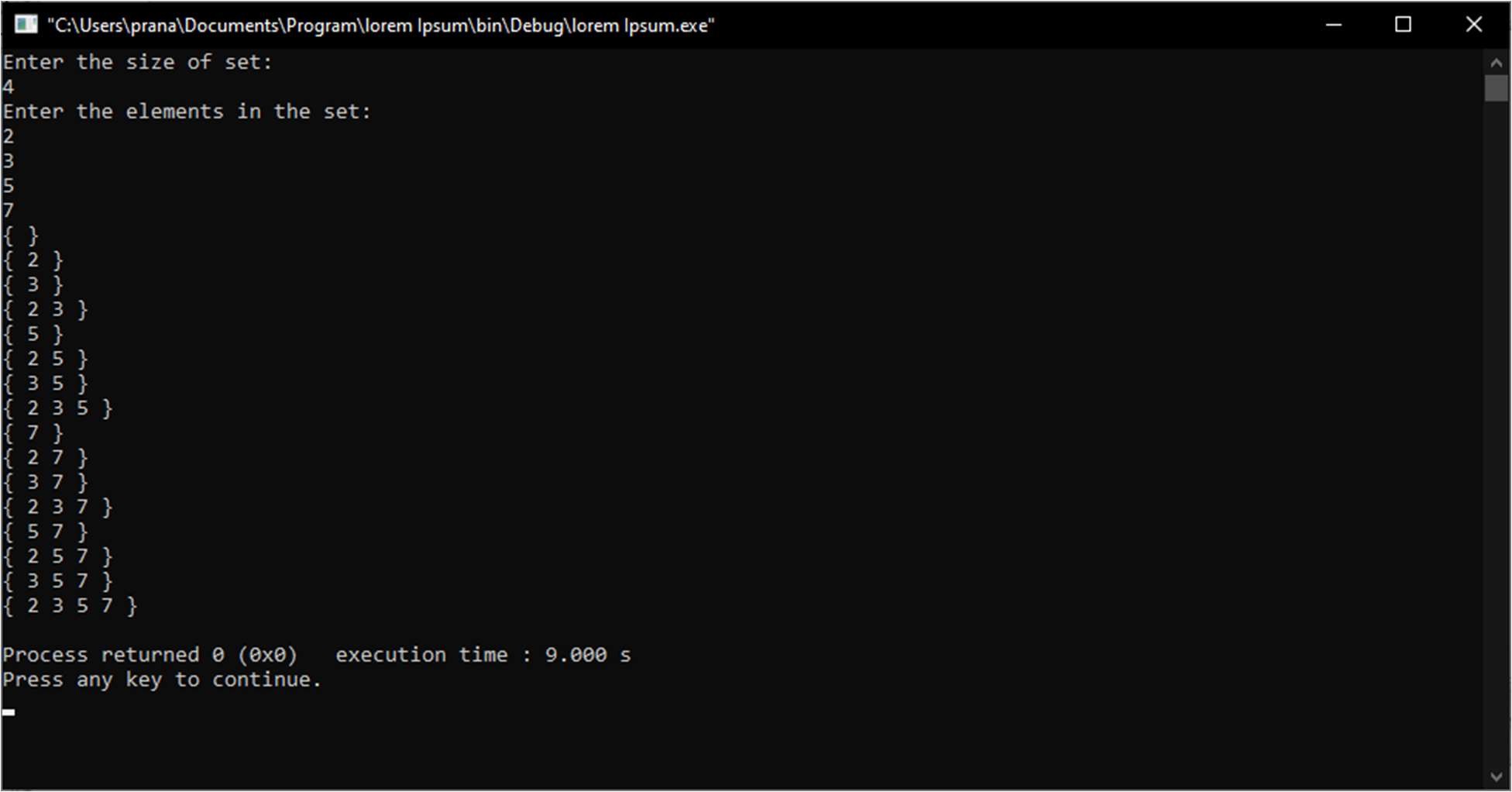
}

printPowerSet(set, n); getchar();

return 0;

}

# OUTPUT



**PRACTICAL NO. 6**

**Aim :- Write a program to display the Truth Table for AND, OR, NOT.**

**Program:-**

#include<stdio.h>

int main()

{

int a[2][2],b[2][2],c[2];

int i,j; for(i=0;i<=1;i++)

{

for(j=0;j<=1;j++)

{

a[i][j]=(i&&j);

b[i][j]=(i||j);

}

}

for(i=0;i<=1;i++)

{ c[i]=(!i);

}

printf("\nThe Truth Table for AND Gate( && ) is..\n"); printf(" A B : C=A&&B\n");

for(i=0;i<=1;i++)

{

for(j=0;j<=1;j++)

{

printf(" %d %d : %d\n",i,j,a[i][j]);

}

}

printf("\nThe Truth Table for OR Gate( || ) is..\n"); printf(" A B : C=A||B\n"); for(i=0;i<=1;i++)

{

for(j=0;j<=1;j++)

{

printf(" %d %d : %d\n",i,j,b[i][j]);

}

}

printf("\nThe Truth Table for NOT Gate (!) is..\n"); printf(" A : B = !A\n");

for(i=0;i<=1;i++)

{

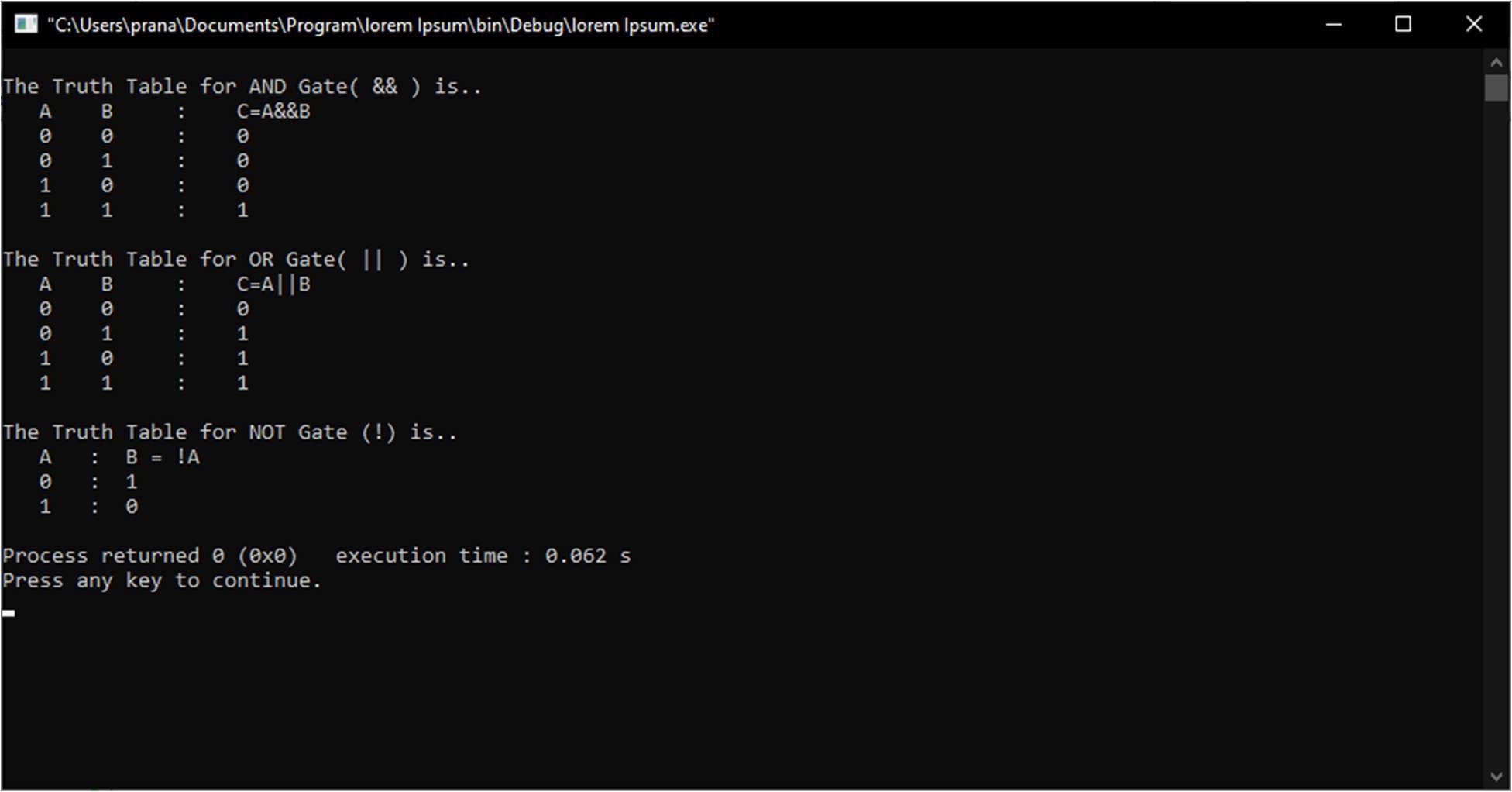
printf(" %d : %d\n",i,c[i]);

}

return 0;

}

# OUTPUT



**PRACTICAL NO. 7**

**Aim :- Write a program to find cartesian product of two sets.**

**Program:-**

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

{

int a[50],b[50],c[50],i,s1,s2,j,k;

printf("Enter number of elements in set 1\n"); scanf("%d",&s1);

printf("Enter number of elements in set 2\n"); scanf("%d",&s2);

printf("Enter elements of set 1\n"); for(i=0;i<s1;i++)

{

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

}

printf("Enter elements of set 2\n"); for(i=0;i<s2;i++)

{

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

}

printf("cartesian product="); printf("{"); for(i=0;i<s1;i++)

{

for(j=0;j<s2;j++)

{

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

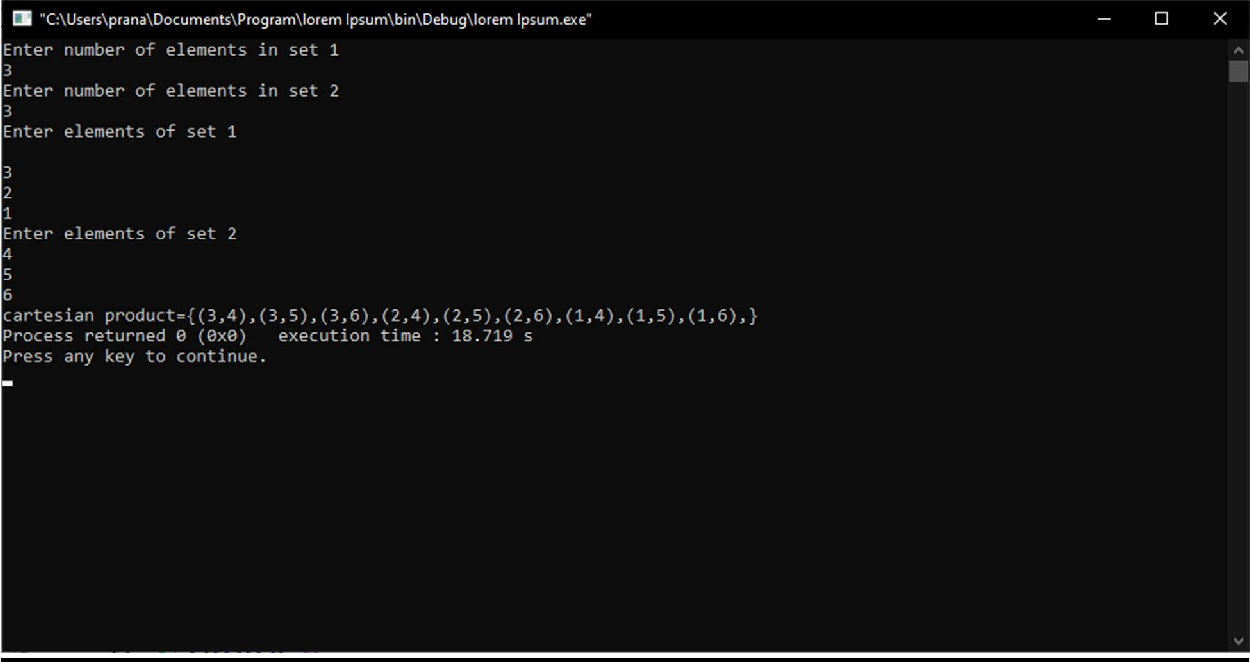
}

}

printf("}"); return 0;

}

# OUTPUT



**PRACTICAL NO. 8**

**Aim :- Write a program for minimum cost spanning tree.**

**Program:-**

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

#define V 5

int minKey(int key[], bool mstSet[])

{

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (mstSet[v] == false && key[v] < min) min = key[v], min\_index = v;

return min\_index;

}

int printMST(int parent[], int graph[V][V])

{

printf("Edge \tWeight\n"); for (int i = 1; i < V; i++)

printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);

}

void primMST(int graph[V][V])

{

int parent[V]; int key[V]; bool mstSet[V];

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

key[i] = INT\_MAX, mstSet[i] = false;

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < V - 1; count++) { int u = minKey(key, mstSet);

mstSet[u] = true;

for (int v = 0; v < V; v++)

if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) parent[v] = u, key[v] = graph[u][v];

}

printMST(parent, graph);

}

int main()

{

int graph[V][V] = { { 0, 2, 0, 6, 0 },

{ 2, 0, 3, 8, 5 },

{ 0, 3, 0, 0, 7 },

{ 6, 8, 0, 0, 9 },

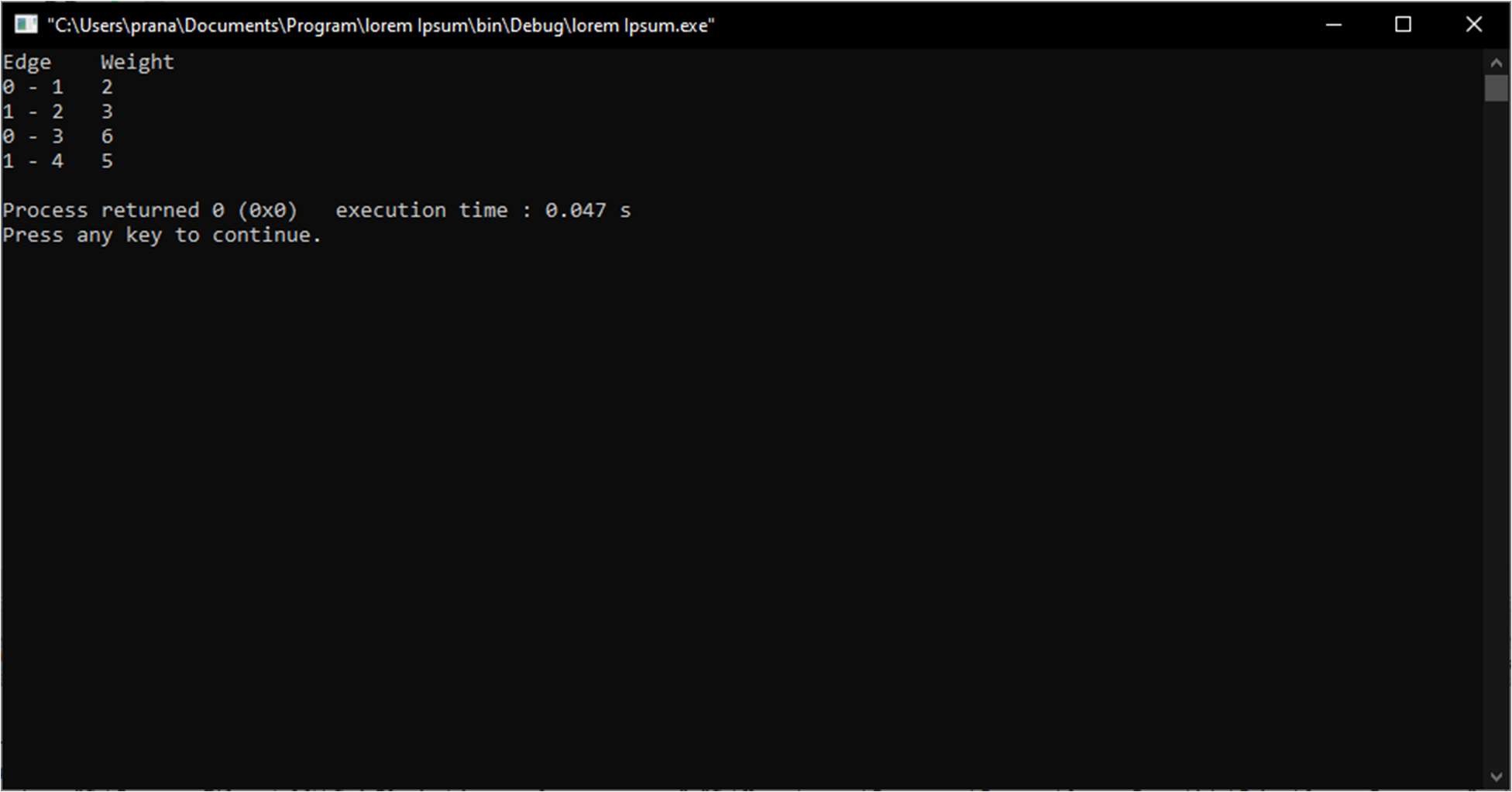
{ 0, 5, 7, 9, 0 } };

primMST(graph);

return 0;

}

# OUTPUT



**PRACTICAL NO. 9**

**Aim :- Write a program for finding the shortest path in a graph.**

**Program:-**

#include<stdio.h> #include<conio.h> #define INFINITY 9999

#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode); int main()

{

int G[MAX][MAX],i,j,n,u; printf("Enter no. of vertices:"); scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

printf("\nEnter the starting node:"); scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX]; int visited[MAX],count,mindistance,nextnode,i,j;

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

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

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

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

{

distance[i]=cost[startnode][i]; pred[i]=startnode; visited[i]=0;

}

distance[startnode]=0; visited[startnode]=1; count=1;

while(count<n-1)

{

mindistance=INFINITY;

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

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i]; nextnode=i;

}

//check if a better path exists through nextnode visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

count++;

}

distance[i]=mindistance+cost[nextnode][i]; pred[i]=nextnode;

}

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

if(i!=startnode)

{

printf("\nDistance of node %d = %d",i,distance[i]); printf("\nPath=%d",i);

j=i;

do

{

j=pred[j]; printf("<-%d",j);

}while(j!=startnode);

}

}

# OUTPUT

