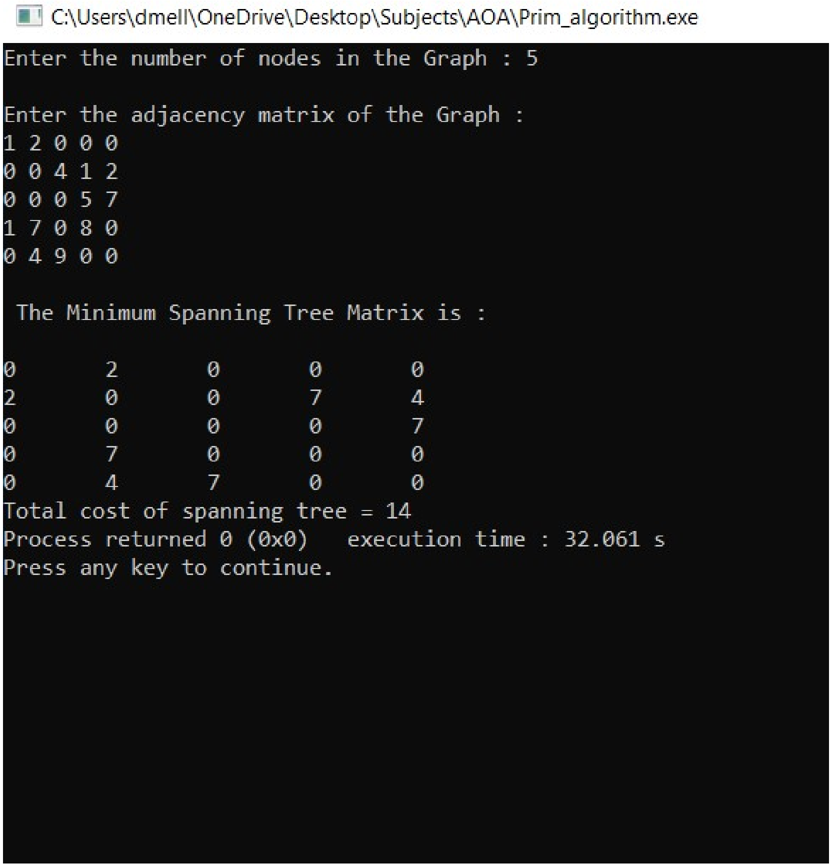
Name: Vanessa D’mello
Roll No. 8863
Branch: SE Computers A(Batch A)
Experiment 8: Prim’s Algorithm

```
#include<stdio.h>
#include<stdlib.h>
#define infinity 9999
#define MAX 20

int prims(int G[MAX][MAX],int spanning[MAX][MAX],int n)
{
	int cost[MAX][MAX];
	int u,v,min_distance,distance[MAX],from[MAX];
	int visited[MAX],no_of_edges,i,min_cost,j;


	//create cost[][] matrix,spanning[][]
	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];
				spanning[i][j]=0;
		}

	//initialize visited[],distance[] and from[]
	distance[0]=0;
	visited[0]=1;

	for(i=1;i<n;i++)
	{
		distance[i]=cost[0][i];
		from[i]=0;
		visited[i]=0;
	}

	min_cost=0;		//cost of spanning tree
	no_of_edges=n-1;		//no. of edges to be added

	while(no_of_edges>0)
	{
		//find the vertex at minimum distance from the tree
		min_distance=infinity;
		for(i=1;i<n;i++)
			if(visited[i]==0&&distance[i]<min_distance)
			{
				v=i;
				min_distance=distance[i];
```

```
            }

        u=from[v];

        //insert the edge in spanning tree
        spanning[u][v]=distance[v];
        spanning[v][u]=distance[v];
        no_of_edges--;
        visited[v]=1;

        //updated the distance[] array
        for(i=1;i<n;i++)
            if(visited[i]==0&&cost[i][v]<distance[i])
            {
                distance[i]=cost[i][v];
                from[i]=v;
            }

        min_cost=min_cost+cost[u][v];
    }

    return(min_cost);
}

int main()
{
    int i,j,total_cost;
    int G[MAX][MAX],spanning[MAX][MAX],n;

    printf("Enter the number of nodes in the Graph : ");
    scanf("%d",&n);


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

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);

    total_cost=prims(G,spanning,n);
    printf("\n The Minimum Spanning Tree Matrix is : \n");

    for(i=0;i<n;i++)
    {
        printf("\n");
        for(j=0;j<n;j++)
            printf("%d\t",spanning[i][j]);
    }
    printf("\nTotal cost of spanning tree = %d",total_cost);
    return 0;
}
```

Output:



C:\Users\dmell\OneDrive\Desktop\Subjects\AOA\Prim_algorithm.exe

```
Enter the number of nodes in the Graph : 5

Enter the adjacency matrix of the Graph :
1 2 0 0 0
0 0 4 1 2
0 0 0 5 7
1 7 0 8 0
0 4 9 0 0

 The Minimum Spanning Tree Matrix is :

0       2       0       0       0
2       0       0       7       4
0       0       0       0       7
0       7       0       0       0
0       4       7       0       0
Total cost of spanning tree = 14
Process returned 0 (0x0)   execution time : 32.061 s
Press any key to continue.
```

Postlab:

1. Do you get same spanning tree when you apply kruskal and prim's algorithm on same graph? Justify your answer

- If the edge weights in your graph are all different from each other, then your graph has a unique minimum spanning tree, so Kruskal's and Prim's algorithms are guaranteed to return the same tree.
- If the edge weights in your graph are not all different , then neither algorithm is necessarily deterministic. They both have steps of the form "choose the lowest-weight edge that satisfies some condition" that might yield ambiguous results. For example, in the extreme case where all edges have the same weight, either algorithm could conceivably return any of the graph's spanning trees. That is, Prim's algorithm might yield a different minimum spanning tree than Kruskal's algorithm in this case.