Experiment No. 7

/* A pizza shop receives multiple orders from several locations. Assume that one pizza boy is tasked with delivering pizzas in nearby locations, which is represented using a graph. The time required to reach from one location to another represents node connections. Solve the problem of delivering a pizza to all customers in the minimum time. Use appropriate data structures.*/

```
#include <stdio.h>

#define MAX 20

#define INF 999

int a[MAX][MAX], visited[MAX];

int v, e; // v = number of locations, e = number of edges


void input()

{

  int i, j, from, to, time;

  printf("Enter the number of locations (v): ");

  scanf("%d", &v);

  // Initialize adjacency matrix and visited array

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

{

    visited[i] = 0;

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

   {

      a[i][j] = INF; // no direct route

    }

  }


  printf("\nEnter the number of edges (connections) (e): ");
```

```
    scanf("%d", &e);

  for(i = 0; i < e; i++)
    {
        printf("Enter the end locations of edge %d: ", i + 1);
        scanf("%d %d", &from, &to);
        printf("Enter the travel time for this edge: ");
        scanf("%d", &time);
        a[from-1][to-1] = time;
        a[to-1][from-1] = time; // undirected graph
    }
}

void display()
 {
    int i, j;
    printf("\nAdjacency Matrix (Travel Times):\n");
    for(i = 0; i < v; i++)
    {
        for(j = 0; j < v; j++)
        {
            if(a[i][j] == INF)
                printf("INF ");
            else
                printf("%d ", a[i][j]);
        }
```

```
        printf("\n");
    }
}



void minimum_Delivery_Time()
{
    int i, j, count;
    int total_Time = 0, min, from = 0, to = 0;
    visited[0] = 1; // start from first location
    printf("\nMinimum Delivery Routes (Using Prim's Algorithm):\n");
    for(count = 0; count < v - 1; count++)
     {
        min = INF;
        for(i = 0; i < v; i++)
          {
           if(visited[i])
               {
              for(j = 0; j < v; j++)
                 {
                 if(!visited[j] && a[i][j] < min)
                     {
                    min = a[i][j];
                    from = i;
                    to = j;
```

```
            }
        }
    }
}


 visited[to] = 1; // mark newly added location
    total_Time += min;
    printf("Deliver from Location %d -> Location %d with travel time: %d\n", from+1, to+1,
min);
  }
  printf("\nMinimum Total Delivery Time: %d\n", total_Time);
}

int main()
 {
  int choice;
  do {
    printf("\n==========PIZZA DELIVERY MIN TIME=================\n");
    printf("1. INPUT LOCATIONS AND EDGES\n2. DISPLAY MATRIX\n3. MINIMUM DELIVERY
TIME\n4. EXIT\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);



    switch(choice)
```

```
        {
          case 1:
             input();
             break;
          case 2:
             display();
             break;
          case 3:
             minimum_Delivery_Time();
             break;
          case 4:
             printf("Exiting program...\n");
             break;
          default:
             printf("Invalid choice!\n");
        }
    } while(choice != 4);
    return 0;
}
```

Output:-

```
==========PIZZA DELIVERY MIN TIME=================
1. INPUT LOCATIONS AND EDGES
2. DISPLAY MATRIX
3. MINIMUM DELIVERY TIME
4. EXIT
Enter your choice: 1
```

Enter the number of locations (v): 6

Enter the number of edges (connections) (e): 12

Enter the end locations of edge 1: 3 2

Enter the travel time for this edge: 5

Enter the end locations of edge 2: 2 6

Enter the travel time for this edge: 4

Enter the end locations of edge 3: 1 6

Enter the travel time for this edge: 3

Enter the end locations of edge 4: 6 5

Enter the travel time for this edge: 4

Enter the end locations of edge 5: 3 7

Enter the travel time for this edge: 2

Enter the end locations of edge 6: 6 5

Enter the travel time for this edge: 1

Enter the end locations of edge 7: 6 8

Enter the travel time for this edge: 7

Enter the end locations of edge 8: 5 9

Enter the travel time for this edge: 1

Enter the end locations of edge 9: 3 9

Enter the travel time for this edge: 9

Enter the end locations of edge 10: 6 8

Enter the travel time for this edge: 4

Enter the end locations of edge 11: 2 6

Enter the travel time for this edge: 6

Enter the end locations of edge 12: 5 1

Enter the travel time for this edge: 5

==========PIZZA DELIVERY MIN TIME==========

1. INPUT LOCATIONS AND EDGES

2. DISPLAY MATRIX

3. MINIMUM DELIVERY TIME

4. EXIT

Enter your choice: 2

Adjacency Matrix (Travel Times):

INF INF INF INF 5 3

INF INF 5 INF INF 6

INF 5 INF INF INF INF

INF INF INF INF INF INF

5 INF INF INF INF 1

3 6 INF INF 1 INF

==========PIZZA DELIVERY MIN TIME==========

1. INPUT LOCATIONS AND EDGES

2. DISPLAY MATRIX

3. MINIMUM DELIVERY TIME

4. EXIT

Enter your choice: 3

Minimum Delivery Routes (Using Prim's Algorithm):

Deliver from Location 1 -> Location 6 with travel time: 3

Deliver from Location 6 -> Location 5 with travel time: 1

Deliver from Location 6 -> Location 2 with travel time: 6

Deliver from Location 2 -> Location 3 with travel time: 5

Deliver from Location 2 -> Location 3 with travel time: 999

Minimum Total Delivery Time: 1014

==========PIZZA DELIVERY MIN TIME==========

1. INPUT LOCATIONS AND EDGES

2. DISPLAY MATRIX

3. MINIMUM DELIVERY TIME

4. EXIT

Enter your choice: 4

Exiting program...

=== Code Execution Successful ===