Assignment 11

NAME: SHISHU

REG: 2020CA089

1. Write a C program to implement Dijkstra's algorithm and find the shortest paths from source (A) to all vertices in the given graph

```
// Dijkstra's Algorithm


#include <stdio.h>

#define INFINITY 9999
#define MAX 10


void Dijkstra(int Graph[MAX][MAX], int n, int start);


void Dijkstra(int Graph[MAX][MAX], int n, int start) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;


  // Creating cost matrix
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (Graph[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = Graph[i][j];


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

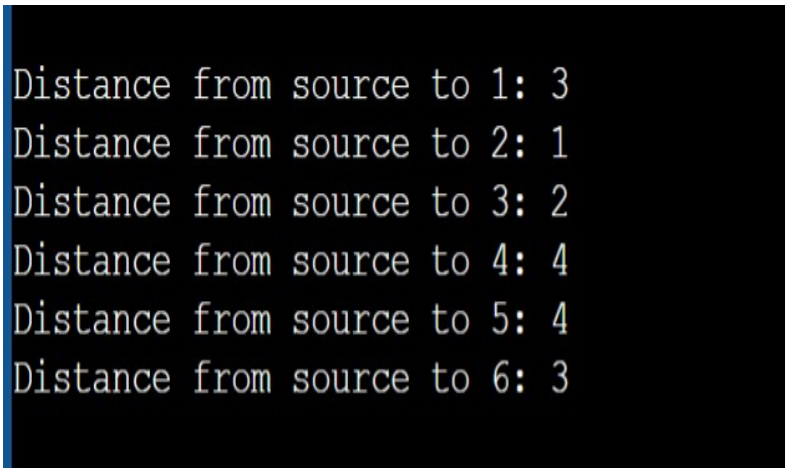
```
  distance[i] = cost[start][i];
  pred[i] = start;
  visited[i] = 0;
}

distance[start] = 0;
visited[start] = 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;
    }

  visited[nextnode] = 1;
  for (i = 0; i < n; i++)
    if (!visited[i])
      if (mindistance + cost[nextnode][i] < distance[i]) {
        distance[i] = mindistance + cost[nextnode][i];
        pred[i] = nextnode;
      }
  count++;
```

```
  }

  // Printing the distance
  for (i = 0; i < n; i++)
    if (i != start) {
      printf("\nDistance from source to %d: %d", i, distance[i]);
    }
}
int main() {
  int Graph[MAX][MAX], i, j, n, u;
  n = 7;

  Graph[0][0] = 0;
  Graph[0][1] = 0;
  Graph[0][2] = 1;
  Graph[0][3] = 2;
  Graph[0][4] = 0;
  Graph[0][5] = 0;
  Graph[0][6] = 0;

  Graph[1][0] = 0;
  Graph[1][1] = 0;
  Graph[1][2] = 2;
  Graph[1][3] = 0;
  Graph[1][4] = 0;
  Graph[1][5] = 3;
```

```
Graph[1][6] = 0;


Graph[2][0] = 1;
Graph[2][1] = 2;
Graph[2][2] = 0;
Graph[2][3] = 1;
Graph[2][4] = 3;
Graph[2][5] = 0;
Graph[2][6] = 0;

Graph[3][0] = 2;
Graph[3][1] = 0;
Graph[3][2] = 1;
Graph[3][3] = 0;
Graph[3][4] = 0;
Graph[3][5] = 0;
Graph[3][6] = 1;

Graph[4][0] = 0;
Graph[4][1] = 0;
Graph[4][2] = 3;
Graph[4][3] = 0;
Graph[4][4] = 0;
Graph[4][5] = 2;
Graph[4][6] = 0;
```

```
  Graph[5][0] = 0;
  Graph[5][1] = 3;
  Graph[5][2] = 0;
  Graph[5][3] = 0;
  Graph[5][4] = 2;
  Graph[5][5] = 0;
  Graph[5][6] = 1;

  Graph[6][0] = 0;
  Graph[6][1] = 0;
  Graph[6][2] = 0;
  Graph[6][3] = 1;
  Graph[6][4] = 0;
  Graph[6][5] = 1;
  Graph[6][6] = 0;

  u = 0;
  Dijkstra(Graph, n, u);
  return 0;
}
```

Output



2. Write a C program to implement coin problem (find minimum number of coin) using greedy approach.

```
#include<stdio.h>
#define max  100

//arr - will have list of needed coins
int ans[max];

int findMinCoins(int coins[], int size,  int value)
{
   int i, count = 0;

   for(i = 0; i < size; i++)
   {
      //take as much from coins[i]
      while(value >= coins[i])
      {
```

```
            //after taking the coin, reduce the value.
            value -= coins[i];
            ans[count] = coins[i];
            count++;
        }
        if(value == 0)
            break;

    }

    return count;
}

int main()
{
    int coins[] = {25,20,10,5};
    int value = 105, i;

    //find the size of the coins array
    int size = sizeof(coins)/sizeof(coins[0]);

    int MinCount = findMinCoins(coins,size,value);

    printf("Total Coins Needed = %d\n",MinCount);

    printf("Coins are:\t");
```

```
    for(i = 0; i < MinCount; i++)
        printf("%d ", ans[i]);

    return 0;
}
```

Output