Roll - 200050006
Ques - 3:

In this question, it was asked to implement the Dijkstra Algorithm to find the shortest path to all vertices from a single source.

Before proceeding the step by step process for implementing the algorithm, let us consider some essential characteristics of Dijkstra's algorithm;

1) Basically, the Dijkstra's algorithm begins from the node to be selected, the source node, and it examines the entire graph to determine the shortest path among that node and all the other nodes in the graph.

2) The algorithm maintains the track of the currently recognized shortest distance from each node to the source code and updates these values if it identifies another shortest path.

3) Once the algorithm has determined the shortest path amid the source code to another node, the node is marked as "visited" and can be added to the path.

4) This process is being continued till all the nodes in the graph have been added to the path, as this way, a path gets created that connects the source node to all the other nodes following the plausible shortest path to reach each node.

Step By Step Algorithm :

1) The very first step is to mark all nodes as unvisited,

2) Mark the picked starting node with a current distance of 0 and the rest nodes with infinity,

3) Now, fix the starting node as the current node,

4) For the current node, analyse all of its unvisited neighbours and measure their distances by adding the current distance of the current node to the weight of the edge that connects the neighbour node and current node,

5) Compare the recently measured distance with the current distance assigned to the neighbouring node and make it as the new current distance of the neighbouring node,

6) After that, consider all of the unvisited neighbours of the current node, mark the current node as visited,

7) If the destination node has been marked visited then stop, an algorithm has ended, and

8) Else, choose the unvisited node that is marked with the least distance, fix it as the new current node, and repeat the process again from step 4.

In the original code, I have compared all the registers with the actual variables used for implement Dijkstra's Algorithm in C++

The code for the Dijkstra function to whose variables I have compared the registers is mentioned below:

C++ Implementation :

```
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;
    }
    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++;
  }
}
```

Assumptions :

1) All the arrays used for solving and the adjacency matrix for the graph as well as the distance array to store all the distances is assumed to be an byte array. So the maximum distance of any node from source node should be less than 256 because it is the maximum a single byte can store.
2) I have assumed the counting of nodes start from 0, 1, 2, 3, … and goes on.
3) In the screenshot I have assumed 6 vertices and 0 as the source vertex.