**NAME: SYED ARSALAN KHAN**

**REG NO: FA19-BCE-010**

**DSA LAB 13**

**POST LAB**

In this report I am going to explain the implementation of depth first algorithm which I coded in lab 13 in lab task 2. We were given a 10 by 10 matrix in which some places node weights were given from the graph in lab manual and other places were assigned -1. By looking at the graph we had to implement a depth first algorithm to go from source to destination and find the cost.

To find the path between Source and destination a void return type function was given in which four parameters were passed. The first parameter was a pointer to adjacency matrix of a graph, size of the matrix and the source and destination. In depth first search we start from a node and put it into a stack and remove it and then move to its neighbors and put then into stack and remove one of the neighbor and then move that nodes neighbor in stack and remove one and do this until all nodes are traversed and stack becomes empty.

In the code first we declared an array visited with size of 10 and assigned 0 in the array. Then two variables current visited and current exploring were declared and assigned src. They were given the value of source. Path cost variable was assigned 0. A while loop is executed in which visited array that we assigned 0 will be assigned 1 when the source node is visited indicatiog this node is visited. Now if current visiting variable is equal to destination then path is founded and we will print path and cost. And will empty the stack. Else if current visiting variable is not equal to destination then assigning current visiting variable 0 and a loop will execute as long as current visiting is less than size of matrix. When the loop executes and if current visiting variable becomes equal to size indicating no unvisited nodes to visit. We will go back to previous nodes. Else structure temp’s vertex number will be assigned the value of current exploring. Structure temp’s cost to visit will be assigned the value of path cost. Push out the top node from stack current exploring will be assigned current visiting.

SO my code works like this. It start from source and checks whether it is the destination. If not it moves to itts neighbor. And the weight is added to cost. Now it checks again for destination. If still it has not reached its destination it moves to its neighbor. If it reached a node with no neighbors and destination is still not reached it moves backward. So, in this way it find’s its destination and returns the cost.