```
//7A -


// C++ implementation of the approach
#include <bits/stdc++.h>
using namespace std;

// adjacency matrix
vector<vector<int> > adj;

// function to add edge to the graph
void addEdge(int x, int y)
{
      adj[x][y] = 1;
      adj[y][x] = 1;
}

// function to perform DFS on the graph
void dfs(int start, vector<bool>& visited)
{

      // Print the current node
      cout << start << " ";

      // Set current node as visited
      visited[start] = true;

      // For every node of the graph
      for (int i = 0; i < adj[start].size(); i++) {

            // If some node is adjacent to the current node
            // and it has not already been visited
            if (adj[start][i] == 1 && (!visited[i])) {
                  dfs(i, visited);
            }
      }
}

int main()
{
      // number of vertices
      int v = 5;

      // number of edges
      int e = 4;

      // adjacency matrix
      adj = vector<vector<int> >(v, vector<int>(v, 0));

      addEdge(0, 1);
      addEdge(0, 2);
      addEdge(0, 3);
      addEdge(0, 4);

      // Visited vector to so that
      // a vertex is not visited more than once
      // Initializing the vector to false as no
      // vertex is visited at the beginning
```

```
    vector<bool> visited(v, false);

    // Perform DFS
    dfs(0, visited);
    return 0;
}
```