**GROUP A**

**Title:** Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

1. **BFS(Breadth First Search)**

**Program:**

import java.util.Iterator;

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

Graph g = new Graph(8);

g.AddEdge(1,2);

g.AddEdge(1,5);

g.AddEdge(2,3);

g.AddEdge(2,5);

g.AddEdge(3,4);

g.AddEdge(4,5);

g.AddEdge(4,6);

g.AddEdge(5,4);

g.BFS(1);

}

}

class Graph{

private int NodeNumber;

private LinkedList<Integer>AdjacentNodes[];

Graph(int V){

AdjacentNodes = new LinkedList[V];

for(int i = 0; i<AdjacentNodes.length;i++){

AdjacentNodes[i] = new LinkedList();

}

NodeNumber = V;

}

public void AddEdge(int v, int w){

AdjacentNodes[v].add(w);

}

public void BFS(int s){

boolean visited[] = new boolean[NodeNumber];

for(int i = 0; i<NodeNumber;i++){

visited[i] = false;

}

LinkedList<Integer>queue = new LinkedList();

visited[s] = true;

queue.add(s);

while(queue.size()!=0){

s=queue.poll();

System.out.println("Visiting"+s+"");

Iterator<Integer>i = AdjacentNodes[s].listIterator();

while(i.hasNext()){

int n = i.next();

if(!visited[n]){

visited[n] = true;

queue.add(n);

}

}

}

}

}

**Output:**



1. **DFS(Depth First Search)**

**Program:**

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

public class Main {

public static void main(String[] args) {

Graph g = new Graph(8);

g.AddEdge(1,2);

g.AddEdge(1,5);

g.AddEdge(2,3);

g.AddEdge(2,5);

g.AddEdge(3,4);

g.AddEdge(4,5);

g.AddEdge(4,6);

g.AddEdge(5,6);

g.DFS();

}

}

class Graph{

private int NodeNumber;

private LinkedList<Integer>AdjacentNodes[];

Graph(int V){

AdjacentNodes = new LinkedList[V];

for(int i = 0; i<AdjacentNodes.length;i++){

AdjacentNodes[i] = new LinkedList();

}

NodeNumber = V;

}

public void AddEdge(int v, int w){

AdjacentNodes[v].add(w);

}

void DFSUtil(int v,boolean visited[]){

visited[v] = true;

System.out.println("visiting"+v+"");

Iterator<Integer>i = AdjacentNodes[v].listIterator();

while(i.hasNext()){

int n = i.next();

if(!visited[n]){

DFSUtil(n,visited);

}

}

}

public void DFS(){

boolean visited[] = new boolean[NodeNumber];

for(int i = 0; i<NodeNumber; ++i){

if(visited[i]==false){

DFSUtil(i,visited);

}

}

}

}

print(dfs\_iterative(test\_graph,'A'))

**Output:**

