Student ID: 202112047

IT623 - Lab Assignment 10

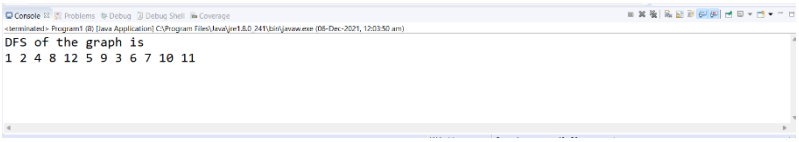
1. Implement BFS on a given graph.

Code:

```
import java.util.*;

public class Program1 {
        int vertices;
        LinkedList<Integer> adjLists[];

        Program1(int v)
   {
      vertices = v;
      adjLists = new LinkedList[v];
      for(int i = 0; i < v; i++)
      {
       adjLists[i] = new LinkedList<>();
      }
   }

        void insertEdge(int src, int dest) {
                adjLists[src].add(dest);
        }

        void DFS(int v, boolean visited[]) {
                visited[v] = true;
                System.out.print(v + " ");

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

```
            while (i.hasNext()) {
                int n = i.next();
                if (!visited[n]) {
                    DFS(n, visited);
                }
            }
        }

        void traverseDFS(int v) {
            boolean visited[] = new boolean[vertices];
            DFS(v, visited);
        }

        public static void main(String args[]) {
            Program1 p = new Program1(14);

            p.insertEdge(1, 2);
            p.insertEdge(1, 3);
            p.insertEdge(2, 4);
            p.insertEdge(2, 5);
            p.insertEdge(3, 6);
            p.insertEdge(3, 7);
            p.insertEdge(4, 8);
            p.insertEdge(5, 9);
            p.insertEdge(7, 10);
            p.insertEdge(7, 11);
            p.insertEdge(8, 12);

            System.out.println("DFS of the graph is ");
            p.traverseDFS(1);
        }
}
```

Output Snapshot:



2. Print largest value of each row in tree.

Code:

```
public class Program2 {

    public static class Node
    {
        int data;
        Node left, right;
    };

    static String string;

    public static Node newNode(int data)
    {
        Node node = new Node();
        node.data = data;
        node.left = node.right = null;
        return (node);
    }

    static void binaryTreeToString(Node root)
```

```
{
    if (root == null)
        return;

    string += (Character.valueOf((char)
        (root.data + '0')));

    if (root.left == null && root.right == null)
        return;

    string += ('(');
    binaryTreeToString(root.left);
    string += (')');


    if (root.right != null)
    {
     string += ('(');
        binaryTreeToString(root.right);
        string += (')');
    }
}

public static void main(String args[])
{
    Node root = newNode(1);
    root.left = newNode(2);
    root.right = newNode(3);
    root.left.left = newNode(4);

    string = "";
    binaryTreeToString(root);
    System.out.println("String is " + string);
```

```
        }

}
```

Output Snapshot: