Experiment 2.2

Studentname: Rohan Jaiswal
UID: 21BCS2856
Branch: BE-CSE
Section/Group: KRG_CC_1-B
Semester: 6th
Date of Performance: 26-02-2024
Subject Name: Advanced Programming lab-2
Subject Code: 21CSP-351

Aim:

1. To Solve the Gray Code Problem
2. To Solve the Is Graph Bipartite Problem

Objective:

- **There is an undirected graph with n nodes, where each node is numbered between 0 and n - 1. You are given a 2D array graph, where graph[u] is an array of nodes that node u is adjacent to. More formally, for each v in graph[u], there is an undirected edge between node u and node v. The graph has the following properties:.**

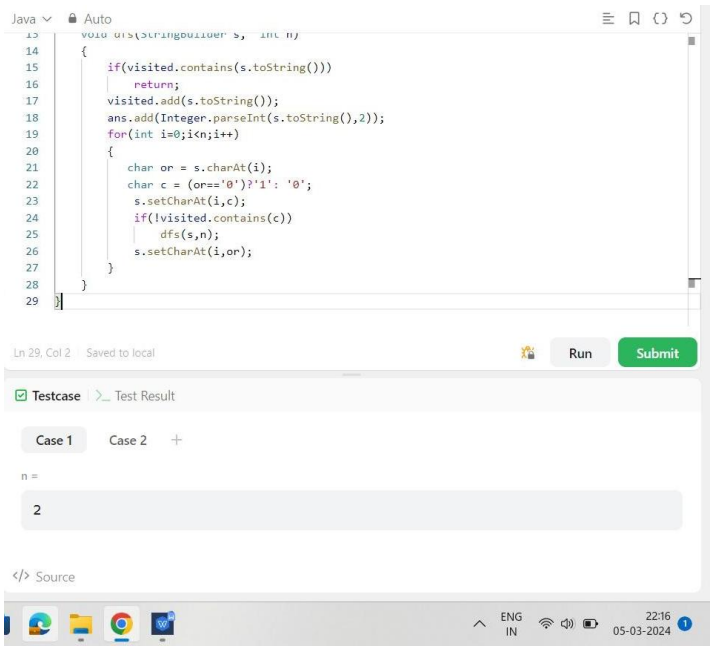
- **An n-bit gray code sequence is a sequence of 2n integers where:Every integer is in the inclusive range [0, 2n - 1],**

Code(A):

```
class Solution
  { List<Integer> ans;
  HashSet<String> visited;
  public List<Integer> grayCode(int n)
     { visited = new HashSet();
     ans = new ArrayList();
     StringBuilder s = new StringBuilder();
     for(int i=0;i<n;i++)
       s.append('0');
     dfs(s,n);
     return ans;
  }
  void dfs(StringBuilder s,  int n)
  {
     if(visited.contains(s.toString()))
        return;
     visited.add(s.toString());
     ans.add(Integer.parseInt(s.toString(),2));
     for(int i=0;i<n;i++)
     {
       char or = s.charAt(i);
       char c = (or=='0')?'1': '0';
        s.setCharAt(i,c); if(!visited.contains(c))
```

```
            dfs(s,n);
        s.setCharAt(i,or);
    }
  }
}
```

Output(A):



Code(B):

```
class Solution {
  public boolean isBipartite(int[][] graph)
    { int col[]=new int[graph.length];
    for(int
      i=0;i<col.length;i++){ col
      [i]=-1;//No color
    }
    Queue<Integer> qe=new LinkedList<>();
    for(int i=0;i<graph.length;i++){
      if(col[i]==-1){//BFS
        qe.add(i);
        col[i]=1;//yellow
        while(!qe.isEmpty()){
          int curr=qe.poll();
          int arr[]=graph[curr];
          for(int
            j=0;j<arr.length;j++){ if(co
            l[graph[curr][j]]==-1){

              col[graph[curr][j]]=1-col[curr];
              qe.add(graph[curr][j]);
```

```
                }
                else
                    if(col[graph[curr][j]]==col[curr]){ r
                    eturn false;
                }
            }
        }
    }
}
}
return true;
}
}
```

Output(B):