Name: Nachiket Jagdale

Class: A3-B1

Roll no.-02

PRACTICAL NO. 8

Aim: Implement Graph Colouring algorithm use Graph colouring concept.

Problem Statement:

A GSM is a cellular network with its entire geographical range divided into

hexadecimal cells. Each cell has a communication tower which connects with mobile

phones within cell. Assume this GSM network operates in different frequency

ranges. Allot frequencies to each cell such that no adjacent cells have same

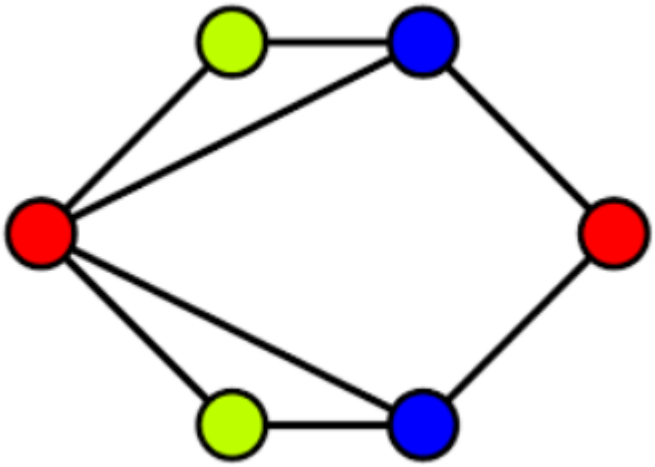
frequency range.

Consider an undirected graph G = (V, E) shown in fig. Find the colour assigned to each node

using Backtracking method. Input is the adjacency matrix of a graph G(V, E), where V is the

number of Vertices and E is the number of edges.

Graph 1:



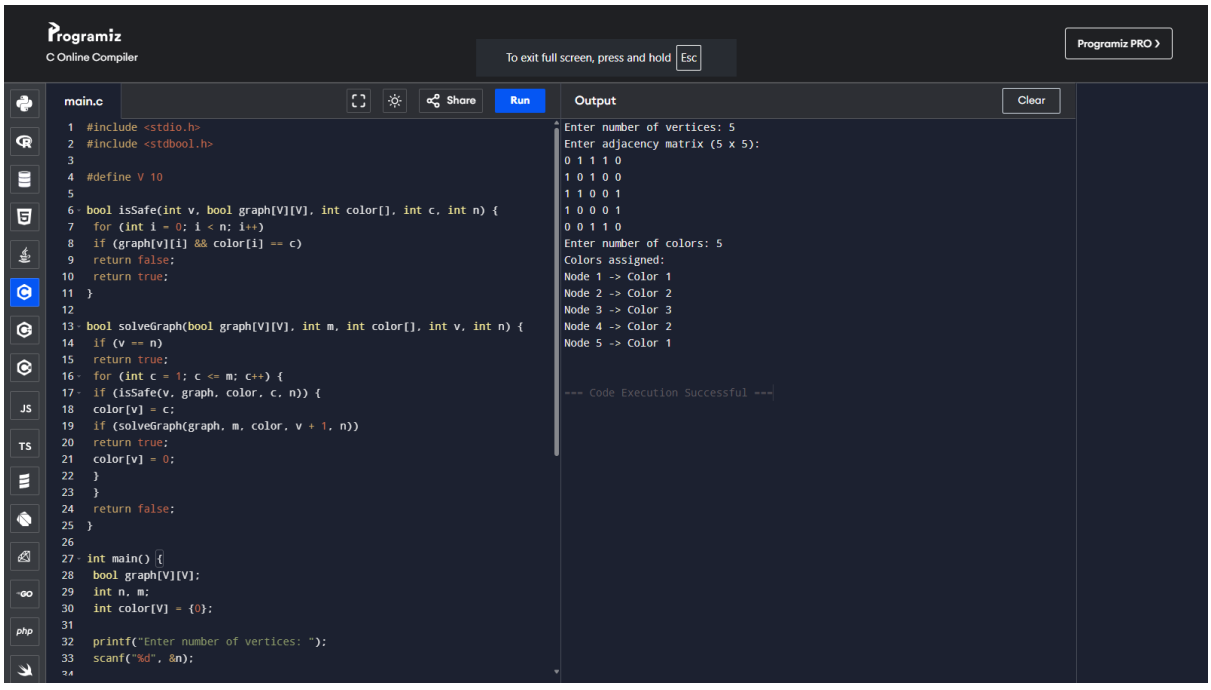
CODE:

```
#include <stdio.h>
#include <stdbool.h>

#define V 10

bool isSafe(int v, bool graph[V][V], int color[], int c, int n) {
 for (int i = 0; i < n; i++)
 if (graph[v][i] && color[i] == c)
 return false;
 return true;
}

bool solveGraph(bool graph[V][V], int m, int color[], int v, int n) {
 if (v == n)
 return true;
 for (int c = 1; c <= m; c++) {
```

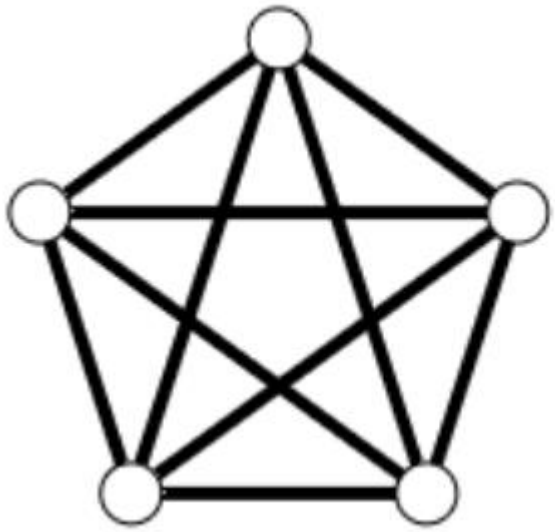
```
 if (isSafe(v, graph, color, c, n)) {
 color[v] = c;
 if (solveGraph(graph, m, color, v + 1, n))
 return true;
 color[v] = 0;
 }
 }
 return false;
}

int main() {
 bool graph[V][V];
 int n, m;
 int color[V] = {0};

 printf("Enter number of vertices: ");
 scanf("%d", &n);

 printf("Enter adjacency matrix (%d x %d):\n", n, n);
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 scanf("%d", &graph[i][j]);

 printf("Enter number of colors: ");
 scanf("%d", &m);
```

```
if (solveGraph(graph, m, color, 0, n)) {
printf("Colors assigned:\n");
for (int i = 0; i < n; i++)
printf("Node %d -> Color %d\n", i + 1, color[i]);
} else {
printf("No solution exists\n");
}

return 0;
}
```

OUTPUT:



GRAPH 2:



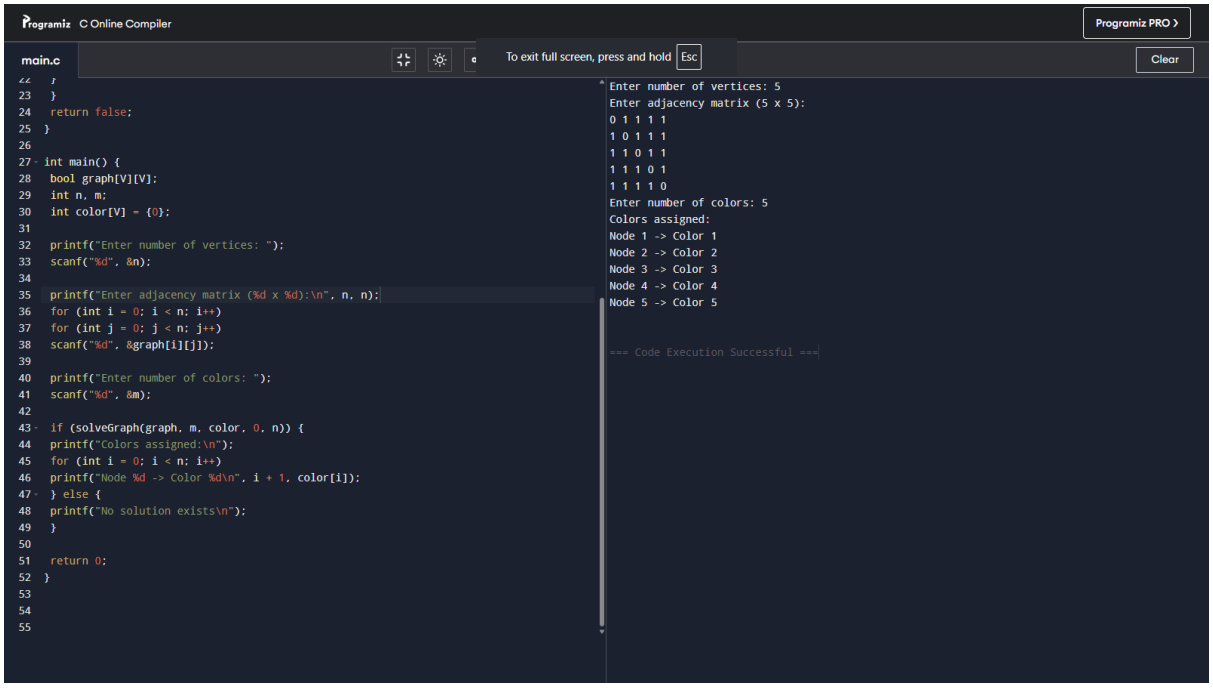
CODE:

```
#include <stdio.h>
#include <stdbool.h>

#define V 10

bool isSafe(int v, bool graph[V][V], int color[], int c, int n) {
 for (int i = 0; i < n; i++)
 if (graph[v][i] && color[i] == c)
 return false;
 return true;
}

bool solveGraph(bool graph[V][V], int m, int color[], int v, int n) {
 if (v == n)
```

```
 return true;
 for (int c = 1; c <= m; c++) {
 if (isSafe(v, graph, color, c, n)) {
 color[v] = c;
 if (solveGraph(graph, m, color, v + 1, n))
 return true;
 color[v] = 0;
 }
 }
 return false;
}

int main() {
 bool graph[V][V];
 int n, m;
 int color[V] = {0};

 printf("Enter number of vertices: ");
 scanf("%d", &n);

 printf("Enter adjacency matrix (%d x %d):\n", n, n);
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 scanf("%d", &graph[i][j]);
```

```
printf("Enter number of colors: ");
scanf("%d", &m);

if (solveGraph(graph, m, color, 0, n)) {
printf("Colors assigned:\n");
for (int i = 0; i < n; i++)
printf("Node %d -> Color %d\n", i + 1, color[i]);
} else {
printf("No solution exists\n");
}

return 0;
}
```

OUTPUT:



Competitive Coding Link:
https://www.geeksforgeeks.org/problems/m-coloring-problem-1587115620/1

CODE:

```
import java.util.*;
class Solution {
    boolean graphColoring(int v, int[][] edges, int m) {
        boolean[][] graph = new boolean[v][v];
        for (int[] e : edges) {
            graph[e[0]][e[1]] = true;
            graph[e[1]][e[0]] = true;
        }
        int[] color = new int[v];
        return solve(0, graph, color, m, v);
    }
    boolean solve(int node, boolean[][] graph, int[] color, int m, int v) {
        if (node == v) return true;
        for (int c = 1; c <= m; c++) {
            if (isSafe(node, graph, color, v, c)) {
                color[node] = c;
                if (solve(node + 1, graph, color, m, v)) return true;
                color[node] = 0;
            }
```

```
        }
        return false;
    }
    boolean isSafe(int node, boolean[][] graph, int[] color, int v,
int c) {
        for (int i = 0; i < v; i++)
            if (graph[node][i] && color[i] == c)
                return false;
        return true;
    }
}
```

OUTPUT: