EXPERIMENT NO: 2 **A N V SREEVISHNU**
DATE: 05/02/2021 **RA1811003010333**

GRAPH COLOURING PROBLEM

Aim: To implement graph colouring problem in AI using C++.

Procedure/Algorithm:

- First, create a list of all the colours that need assigning to pass to.
- Condition is set of not two adjacent vertices should have the same colour.
- Numbers are then passed and each number treated as a vertex is given a colour
- Diagram is then drawn to show the graph that is given as the output.

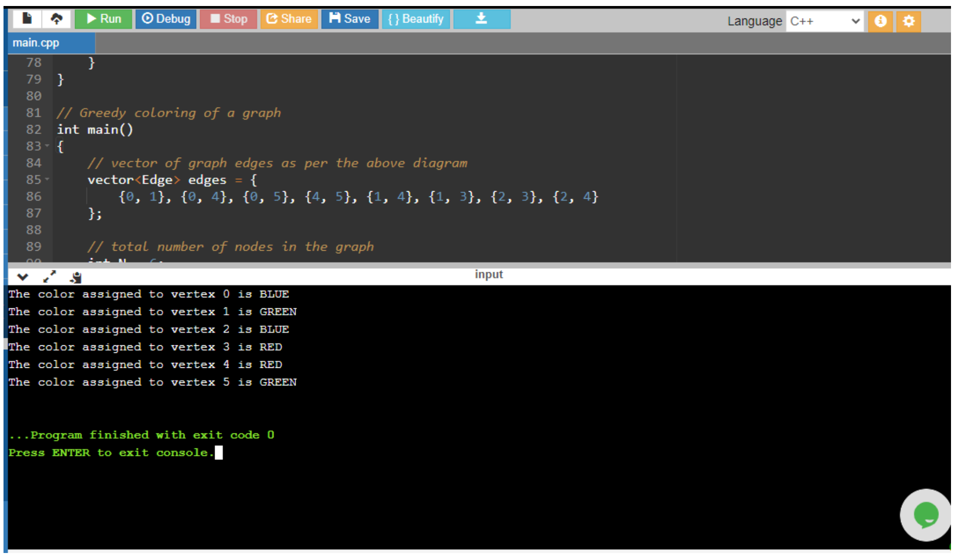
Code:

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <set>
using namespace std;

// Data structure to store a graph edge
struct Edge {
   int src, dest;
};

class Graph
{
public:
   // a vector of vectors to represent an adjacency list
   vector<vector<int>> adjList;

   // Constructor
   Graph(vector<Edge> const &edges, int N)
   {
      // resize the vector to hold `N` elements of type `vector<int>`
      adjList.resize(N);

      // add edges to the undirected graph
      for (Edge edge: edges)
```

```
        {
            int src = edge.src;
            int dest = edge.dest;

            adjList[src].push_back(dest);
            adjList[dest].push_back(src);
        }
    }
};

// Add more colors for graphs with many more vertices
string color[] =
{
    "", "BLUE", "GREEN", "RED", "YELLOW", "ORANGE", "PINK",
    "BLACK", "BROWN", "WHITE", "PURPLE", "VOILET"
};

// Function to assign colors to vertices of a graph
void colorGraph(Graph const &graph, int N)
{
    // keep track of the color assigned to each vertex
    unordered_map<int, int> result;

    // assign a color to vertex one by one
    for (int u = 0; u < N; u++)
    {
        // set to store the color of adjacent vertices of `u`
        set<int> assigned;

        // check colors of adjacent vertices of `u` and store them in a set
        for (int i: graph.adjList[u]) {
            if (result[i]) {
                assigned.insert(result[i]);
            }
        }

        // check for the first free color
        int color = 1;
        for (auto &c: assigned ) {
            if (color != c) {
                break;
            }
            color++;
        }

        // assign vertex `u` the first available color
        result[u] = color;
    }

    for (int v = 0; v < N; v++) {
        cout << "The color assigned to vertex " << v << " is "
            << color[result[v]] << '\n';
    }
}
```

```
// Greedy coloring of a graph
int main()
{
   // vector of graph edges as per the above diagram
   vector<Edge> edges = {
      {0, 1}, {0, 4}, {0, 5}, {4, 5}, {1, 4}, {1, 3}, {2, 3}, {2, 4}
   };

   // total number of nodes in the graph
   int N = 6;

   // build a graph from the given edges
   Graph graph(edges, N);

   // color graph using the greedy algorithm
   colorGraph(graph, N);

   return 0;
}
```

Output:



```
        }
}

// Greedy coloring of a graph
int main()
{
    // vector of graph edges as per the above diagram
    vector<Edge> edges = {
        {0, 1}, {0, 4}, {0, 5}, {4, 5}, {1, 4}, {1, 3}, {2, 3}, {2, 4}
    };

    // total number of nodes in the graph
```

```
The color assigned to vertex 0 is BLUE
The color assigned to vertex 1 is GREEN
The color assigned to vertex 2 is BLUE
The color assigned to vertex 3 is RED
The color assigned to vertex 4 is RED
The color assigned to vertex 5 is GREEN


...Program finished with exit code 0
Press ENTER to exit console.
```

Result: Thus, the implementation of graph colouring problem in AI using C++ has been successfully done.