// A C++ program to implement greedy algorithm for graph coloring

#include <iostream>

#include <list>

using namespace std;

// A class that represents an undirected graph

class Graph

{

int V; // No. of vertices

list<int> \*adj; // A dynamic array of adjacency lists

public:

// Constructor and destructor

Graph(int V) { this->V = V; adj = new list<int>[V]; }

~Graph() { delete [] adj; }

// function to add an edge to graph

void addEdge(int v, int w);

// Prints greedy coloring of the vertices

void greedyColoring();

};

void Graph::addEdge(int v, int w)

{

adj[v].push\_back(w);

adj[w].push\_back(v); // Note: the graph is undirected

}

// Assigns colors (starting from 0) to all vertices and prints

// the assignment of colors

void Graph::greedyColoring()

{

int result[V];

// Assign the first color to first vertex

result[0] = 0;

// Initialize remaining V-1 vertices as unassigned

for (int u = 1; u < V; u++)

result[u] = -1; // no color is assigned to u

// A temporary array to store the available colors. True

// value of available[cr] would mean that the color cr is

// assigned to one of its adjacent vertices

bool available[V];

for (int cr = 0; cr < V; cr++)

available[cr] = false;

// Assign colors to remaining V-1 vertices

for (int u = 1; u < V; u++)

{

// Process all adjacent vertices and flag their colors

// as unavailable

list<int>::iterator i;

for (i = adj[u].begin(); i != adj[u].end(); ++i)

if (result[\*i] != -1)

available[result[\*i]] = true;

// Find the first available color

int cr;

for (cr = 0; cr < V; cr++)

if (available[cr] == false)

break;

result[u] = cr; // Assign the found color

// Reset the values back to false for the next iteration

for (i = adj[u].begin(); i != adj[u].end(); ++i)

if (result[\*i] != -1)

available[result[\*i]] = false;

}

// print the result

for (int u = 0; u < V; u++)

cout << "Vertex " << u << " ---> Color "

<< result[u] << endl;

}

// Driver program to test above function

int main()

{

Graph g1(5);

g1.addEdge(0, 1);

g1.addEdge(0, 2);

g1.addEdge(1, 2);

g1.addEdge(1, 3);

g1.addEdge(2, 3);

g1.addEdge(3, 4);

cout << "Coloring of graph 1 \n";

g1.greedyColoring();

Graph g2(5);

g2.addEdge(0, 1);

g2.addEdge(0, 2);

g2.addEdge(1, 2);

g2.addEdge(1, 4);

g2.addEdge(2, 4);

g2.addEdge(4, 3);

cout << "\nColoring of graph 2 \n";

g2.greedyColoring();

return 0;

}