/\*

Author: Peter O'Donohue

Creation Date: 09/18/17

Modification Date: 09/21/17

Description: FILL IN

\*/

#include <string>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

template <class Object>

class Matrix

{

public:

Matrix(int rows = 0, int cols = 0) : array(rows)

{

for (int i = 0; i < rows; i++)

array[i].resize(cols);

}

void resize(int rows, int cols)

{

array.resize(rows);

for (int i = 0; i < rows; i++)

array[i].resize(cols);

}

const vector<Object> & operator[](int row) const

{

return array[row];

}

vector<Object> & operator[](int row)

{

return array[row];

}

int numrows() const

{

return array.size();

}

int numcols() const

{

return numrows() ? array[0].size() : 0;

}

private:

vector< vector<Object> > array;

};

class Graph

{

public:

Graph(int size); // creates an empty graph with size vertices

void fillGraph(); // fills in the graph from cin

void printGraph(); // prints the graph (for debugging only)

int maxCover(vector<char> order); // returns the maxCover for the

// ordering order

int cover(char vertex, vector<char> order); // returns the cover size for vertex

private:

Matrix<char> adj;

};

Graph::Graph(int size = 3)

{

adj.resize(size, size);

for (int i = 0; i < size; ++i)

for (int j = 0; j < size; ++j)

{

adj[i][j] = ' ';

}

}

void Graph::fillGraph()

{

int numAdj = 0;

int tempRow = 0;

while (adj[tempRow][0] != ' ')

++tempRow;

// cout << "Enter num adj: ";

cin >> numAdj;

adj[tempRow].resize(numAdj); // remove excess space

for (int i = 0; i < numAdj; ++i)

{

// cout << "Enter adjacent: ";

cin >> adj[tempRow][i];

}

}

void Graph::printGraph()

{

for (int i = 0; i < adj.numrows(); ++i)

{

for (int j = 0; j < adj[i].size(); ++j)

{

cout << adj[i][j] << " ";

}

cout << endl;

}

}

int Graph::maxCover(vector<char> order)

{

int max = 0;

int tempMax = 0;

for (int i = 0; i < order.size(); ++i)

{

tempMax = cover(order[i], order);

if (tempMax > max)

max = tempMax;

}

return max;

}

int Graph::cover(char Vertex, vector<char> order)

{

int cover = 0;

return cover;

}

int main()

{

int numGraphs = 0;

int numVertex = 0;

char letter = ' ';

vector<int> maxCover;

vector<char> perm;

Matrix<char> maxOrder;

// cout << "Input number of graphs: ";

cin >> numGraphs;

maxOrder.resize(numGraphs, numGraphs);

for (int i = 0; i < numGraphs; ++i)

{

// cout << "Number of vertex's: ";

cin >> numVertex;

Graph foo(numVertex);

foo.printGraph();

for (int j = 0; j < numVertex; ++j)

{

// cout << "Enter row letter: ";

cin >> letter;

toupper(letter);

perm.push\_back(letter);

foo.fillGraph();

}

foo.printGraph();

sort(perm.begin(), perm.end());

do

{

/\*

for (int i = 0; i < perm.size(); ++i)

{

maxCover.push\_back(foo.maxCover(perm));

}

\*/

} while (next\_permutation(perm.begin(), perm.end()));

}

cout << endl;

system("pause");

return 0;

}