CPSC 2600 Fall 2021 Homework Assignment #6
Due date: Dec 3, 2021 11:59pm

Write a C++ class definition for an abstract data type called `Digraph` that models a directed graph. Vertices are labeled by number from 0 to *n-1* where *n* is the number of vertices in the graph.

Requirements and reminders:

- The use of STL, templates, bitset, and operator overloading is not permitted in any form. The only exception is the deque class used for traversal.
- Programs that fail to compile will receive a grade of 0. You are allowed to re-submit your program before the due date and time, but only the last submission submitted before the due date and time will be graded. **Late submissions are not accepted and result in a zero**.

`Digraph` should have the following data members:

- `int** matrix`: a two-dimensional array for storing the edges
- `int size`: the number of vertices in the graph
- `bool* visited`: an array of bool used for the traversal functions

Implement the following public member functions.

- A copy constructor
- `Digraph(int n)`: A constructor that creates a graph consisting of *n* vertices and no edges.
- `Digraph(string filename)`: A constructor that creates the graph using the file passed into the function. The format of the file is described later in this document.
- `~Digraph()`: An appropriate destructor.
- `void addEdge(int a, int b)`: Adds the edge (a, b) to the graph.
- `void display() const`: Displays the graph's adjacency matrix to the screen using this format (see examples later):

$$M_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

  - Single space between digits.
  - First row (top) and first column (left) is vertex 0, second row and column is vertex 1, …, last row (bottom) and last column (right) is vertex *n* – 1.
- `void displayDFS(int vertex)`: Displays the result of a depth first search starting at the provided vertex. All indices of *visited* are set to false at the beginning. Use a helper function to do the recursive traversal. When you have a choice between selecting two vertices, pick the vertex with the lower number.
- `Digraph& createBFS(int vertex)`: Returns the resulting spanning tree (i.e. a subgraph of the original graph) of a breadth first search. Also displays the vertices as they are visited.
  - Set all indices of *visited* to false.
  - Start the traversal at the provided vertex.
  - Use the *deque* class from STL as the queue. (More instructions on Canvas)
  - You'll have to dynamically allocate a new Digraph inside the function and return it.
- `int inDegree(int i) const`: Returns the in-degree of vertex i.

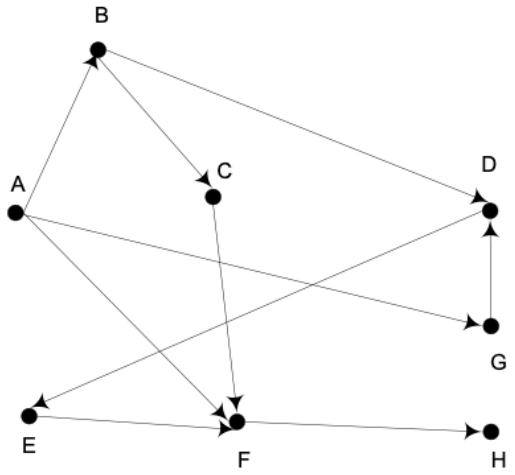
Input File Format

The constructor is responsible for reading in a file corresponding to a graph. The format is as follows:

- The first line contains a single integer indicating how many vertices are present in the graph.
  - You can assume there is at least one vertex.
- All remaining lines contains two integers separated by a comma that indicate a directed edge.
  - The first integer is the tail and the second integer is the head.
- Additional requirements / assumptions:
  - There could be zero or more edges.
  - You can assume that each edge line will contain two different integers (no self-loops).
  - You can assume the same edge will not be repeated in the file (no multiple edges).
- You can assume that the file exists.
- You cannot assume that the graph is connected.
  - In `createBFS` and `displayDFS`, only nodes reachable from the provided vertex are printed out.

As an example, the following graph is represented by the sample file:

```
Index:  0 1 2 3 4 5 6 7
Letter: A B C D E F G H
```




```
8
0,1
0,6
0,5
1,2
1,3
2,5
3,4
4,5
5,7
6,3
```

Submitting your Program

On cs1, run the following script in the directory with your program:

`/home/fac/hkong/submit/cpsc2600/hw6_submit`

Code Organization

Your solution will involve two files for this assignment:

- The class definition must be stored in a file called '`digraph.h`'.
- The class implementation must be stored in a file called '`digraph.cpp`'. You can add extra private member functions as needed.
- A simple driver that tests your Digraph class called `'hw6.cpp'`

At the very least, `hw6.cpp` should carry out the following steps:

1. Create a directed graph using the specified input file.
2. Display the adjacency matrix.
3. Display a breadth first search starting at vertex 0.
4. Create and display a breadth first search starting at vertex 0.
5. Display the adjacency matrix of the spanning tree from Step 4.
6. Display the in-degree of all the vertices.

Sample input files are on Canvas. These sample files and tests are NOT sufficient to fully test your program. You will need to make additional graph files and test cases for complete testing.

Execution example

```
Original graph:

0 1 0 0 0 1 1 0
0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0

Deg-(3) = 2
Deg-(5) = 3

DFS visit order starting at 0:
0 1 2 5 7 3 4 6

BFS visit order starting at 0:
0 1 5 6 2 3 7 4

BFS Spanning tree:

0 1 0 0 0 1 1 0
0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```