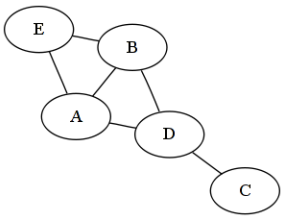
CSCI 2110 Computer Science III
Data Structures and Algorithms
ASSIGNMENT NO. 5

Date Given: Wednesday, November 29, 2017
Due: Tuesday, December 12, 2017, 11.55 p.m.

You are to write a program to read a text file containing information about an undirected, unweighted graph, create an adjacency matrix representation, and traverse the graph using either breadth-first or depth-first search methods.

Follow these steps:

1. Read a text file (specified by the user – don't hard code!) that contains the number of vertices on the first line followed by the edges. You may assume that the vertices are represented by Strings A, B, C, etc. For instance, for the following graph



the input text file will be (order of the edges doesn't matter):

```
5
E  A
E  B
A  B
A  D
B  D
D  C
```

You may also assume that the input file is error-free (that is, it will have the right number of entries and the appropriate values).

2. Next create an adjacency matrix that represents the graph. For example, for the above data, you would create a 2 d array of 5 rows and 5 columns and the adjacency matrix would be:

	A	B	C	D	E
A	0	1	0	1	1
B	1	0	0	1	1
C	0	0	0	1	0
D	1	1	1	0	0
E	1	1	0	0	0

In general, you would create an adjacency matrix of the type:
`int[][] adjMatrix = new int[num][num];`
where num is the value that is read on the first line.

Note: It is easy to convert a String to an integer or a character to an integer. For instance,
`int num = Integer.parseInt(inputFile.nextLine());`
will read the first line in the input text file as an integer.

Then suppose you read the second line as two strings firstVertex and secondVertex:
String firstVertex = inputFile.next();
String secondVertex = inputFile.nextLine();
In the example, firstVertex will be E and secondVertex will be A. With the following statements:
int x = firstVertex.charAt(0)-65;
int y = secondVertex.charAt(0)-65;

x will have the value 4 and y will have the value 0. These are the correct array indices for E and A.

Now all you need to do is to update adjMatrix[x][y] = 1 and adjMatrix[y][x] = 1.

Display the adjacency matrix.

3. Next traverse the graph either using depth first search OR breadth first search method and print the vertices. Print the traversal. ***You can assume any node as the starting vertex.***
The algorithms for depth first search and breadth first search are given below:

Depth First Search (DFS): (recursive algorithm – see lecture notes)
Start at a vertex v1. Put it in the result list.
Pick a neighbor of v1, say v2. Go to v2.
Pick a neighbor of v2, say v3. Go to v3.
Continue and mark each vertex that has been visited. List the marked vertex in the result list.
If you hit a deadend, backtrack to the previous neighbor and pick a neighbor that has not been visited.

Breadth First Search (BFS):
Overview: Start at a vertex v1.
Visit all the neighbors of v1.
Then visit all the neighbors of the neighbors of v1.
Continue until all the vertices are visited.

Algorithm:
Initialize an empty queue and a result list.
Enqueue the first vertex.
while the queue is not empty
Dequeue and list the vertex v in the result list
Enqueue each neighbor of v (if it is not already in the result list or not already in the queue)
end while

What to submit: Source code and sample inputs and outputs in a text file.