CS 601
Programming assignment 1

Due on Wednesday 10/28/2020, 11:59 PM

This is a ***group*** assignment. Each group is supposed to work on the project below together, including the writeup of the solutions. For this assignment, please submit **one** neatly **typed** writeup only **per group**. You need to submit your assignment in the digital repository called "**Project**". You can pick a classmate (**ONLY from your class and NOT another session of CS 413**) to create **a group of size two**. If you prefer to work on the assignment ***individually***, that's also fine.

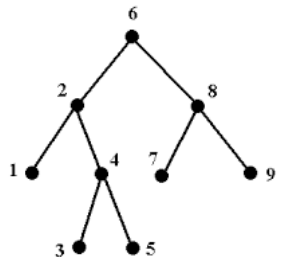
As it appears in the course syllabus, for the assignments, you are expected to turn in the results of your own effort (not the results of another group or person). Even when not explicitly asked, you should justify your answers concisely. Always provide the tightest bound and the most efficient algorithm you can write, even if not explicitly asked.

Consider the following problems and implement their corresponding algorithms **in C++ or Java**:

1) Implement the **Depth first Search** and **Breath first Search** Algorithms in **ONE** C++/Java program (do not submit two programs. You may implement them using two different functions).

Your algorithms should receive an **arbitrary undirected binary tree** and a goal node as inputs (with the format shown below). Then it looks for the goal node and returns success/failure if the node is found/not found.

Example:
Assume we have the following tree given.



The format of the input for the above tree should be as shown in the table below(No alternative format for the input is accepted):

```
6,2
6,8
2,1
2,4
4,3
4,5
8,7
8,9
*
9
#
```

Explanation of the user input: the first eight lines are the eight edges of the tree. For example 6,2 corresponds to the undirected edge (6,2) connecting nodes 6 and 2. If you need to distinguish the root node in your program, you might consider the first number entered to be the root of the tree. For our example, 6 is the root.
Once you are done entering all edges, each in one line, as shown above, you will enter one star. That indicates we are done entering the edges of the tree. The number (label) entered in the line after the * is the target node you need to look for using DFS and BFS algorithms. The last line (#) indicates the end of the input. You need to construct the tree based on the input you've provided to the program.
Now that your algorithm read the input, it should call DFS/BFS algorithm to search through the constructed tree and find the goal node.

2) You need to read the input from an input file called **input.txt**. That helps you test your algorithm for different inputs easier. Also, the grader can run your algorithm using different input values with the help of an input file with the same name.
3) You can print DFS: Success/Failure and BFS: Success/Failure on the screen.

More about submission rules:
To submit the solution to the above assignment, you need to prepare the following files:
1- The source code of your program that includes all files it has including .cpp or .java files.
2- A brief report about the **code** you have written (NO Analysis for the worst-case time complexity of your algorithm is needed for this submission).

- That would be great if you add a standalone executable file of your program if you are able to provide it (optional but recommended)

- For those who submit the project as a group work, you need to include **both names** in the **project report and in the code** otherwise the student submitting the

assignment will be the only owner of the work and we don't accept adding another name later on.

You will get a zero if any of the cases 1 and 2 and the rules mentioned above are missing, or the grader is not able to run your code.