Kerry Ojakian's CSI 33 Class
Due Date: Tuesday October 29 at the start of class

HW #3

General Instructions: Create a folder HW03 in Homework where you put all this. Put written problem 1 into a single PDF document, in this folder. Put the rest of the problems (all programs) into this folder as follows: A single .cpp file containing both problem 2 and 3; a single juypter notebook containing problems 4, 5, 6.

The Assignment

1. From textbook (page 184): Short-Answer Question- 4a and 4c.

   NOTE! - **Not** Programming Exercise 4. Do question 4 at the top of the page (just parts (a) and (c); not part (b)).

2. Write a **template class** for a Stack in C++ so that the stack can contain any data type (not just integers as in class). The Stack should be called exactly `Stack` and the method names should all be the standard names, exactly like in the our class versions, i.e. `push`, `pop`, `top`, `size`.

3. Write a **template class** for a Queue in C++ so that the stack can contain any data type. The Queue should be called exactly `Queue` and the method names should all be the standard names, exactly like in the our class versions, i.e. `enqueue`, `dequeue`, `front`, `size`.

4. Write a Stack class in Python that looks exactly the same (from the outside) as the one we did in class. However, on the inside it should be programmed using a linked list - i.e. use our Python ListNode class. Your pop and push should be O(1) operations.

5. Ch 5, page 184, **Programming Exercise 4**. Write this using the Python Queue class we did in class. The name of the function must be exactly as the problem asks for.

6. Program a simplified version of Programming Exercise 5 (pages 184 to 185). Please read the following details, and note that this exercise is just a slight modification of the program we wrote in class to check if an expression had balanced paretheses when there were three kinds of parentheses (now there are an infinite number of allowed parentheses).

   An example of a **correct** simplified HTML expression:

   ```
   <p> This is a <b> BOLD </b> and correct paragraph </p>
   ```

   An example of an **incorrect** HTML expression:

   ```
   <p> This is a <bdi> BOLD?  </p> WRONG paragraph </bdi>
   ```

   For us an openning tag can be any string of the form "<BLAH>" where *BLAH* can be *any* string whatsoever. Its corresponding closing tag is of exactly the form "</BLAH>", i.e. exactly the same except the forward slash is inserted at the front. For example, if <MorningFriend> is an openning tag, then its corresponding closing tag is </MorningFriend>. Text inbetween the tags is irrelevant. Note that in our simplifed HTML an openning tag may **not** contain anything except the name of the tag (example: <p align ="center"> would actually just be understood by us as a funny openning tag named: p align ="center").

   Create a function, which when executed, should ask the user for a filename, then read in that file, and then print out one of the following two messages: "Correct HTML" if the tags all balance OR "Error in HTML" if the tags do not balance.