CS 0445 Spring 2023
Assignment 2

Online: Monday, February 6, 2023
Due: All source files (both those provided to you and those that you wrote) plus a completed Assignment Information Sheet zipped into a single .zip file and submitted to the proper directory in the submission site by **11:59PM on Monday, February 20, 2023**
Late Due Date: 11:59PM on Saturday, February 25, 2023

Note 1: There will be a 5 point (out of 100) bonus for on-time submission of this assignment. Thus, if you submit on-time and earn all possible points, your score will be 105%.
Note 2: The late penalty for this assignment is only 5 points (out of 100). Thus, if you submit late and earn all possible points, your score will be 95%.
Note 3: Extra credit is available (see end of this document). Extra credit is possible whether submission is on-time or late, and will be up to 10 points (out of 100).

Purpose: Now that we have discussed linked lists, it is a good idea to get some practical experience working with them. In this assignment you will implement a commonly used type in a somewhat uncommon way – using a circular doubly-linked list.

Goal: You should be familiar with the **StringBuilder** class in Java. Recall that the String class in Java produces immutable objects, or, in plain terms, strings that are fixed once they have been created. This can be inefficient if operations such as appending to the end or inserting into the middle of a string are necessary. The StringBuilder class, on the other hand, allows for strings that can be modified. The predefined StringBuilder class is implemented with an underlying array of characters. In this project, you will implement your own class, called MyStringBuilder, which is similar in functionality to the standard StringBuilder, but is implemented using a **linked list of characters** rather than an array.

Details:
I have provided the specifications for the MyStringBuilder class in the file MyStringBuilder.java. Download this file and use this as the starting point for your coding. **Read these specifications over very carefully, paying particular attention to the comments**. Note that your data must be as specified (you **may not** add any additional instance variables) and you must implement your class using your own **circular doubly-linked** list with the inner class CNode as it is written in MyStringBuilder.java.

You may not use any predefined linked list within your class, you may not use any predefined Java String-ish class (such as StringBuilder or StringBuffer) anywhere in your code, and you may not copy any code from the Internet. You may only use the String class when it is either an argument or a return type in one of the MyStringBuilder methods. Furthermore, your methods must act directly on the linked list (so, for example, you may not convert your MyStringBuilder to an array of chars and then perform the method on that array). You must also not do a lot of unnecessary work / extra traversals of your list or arguments in order to perform your methods. For example, in the replace() method, one inefficient approach is to first do a delete() of the specified characters and then do an insert() of the new String. This is not allowed, since it traverses the list twice, when only a single traversal is necessary. As another example, for the copy constructor, you cannot call toString() of the argument and then use the constructor that takes a String, since toString() will traverse the argument list and you will then also need to traverse the resulting String. Think about this issue for all of your methods.

Some of the methods will be relatively simple to implement while some will be more complicated. I recommend working on the methods one at a time, only proceeding to the next one when you are sure that the previous one works correctly. This way, any compilation or logic errors that occur will be fairly easy to locate and will be less difficult to fix. For most of the methods, be very careful to **handle special cases**, as we discussed in lecture. You may find the special cases to be the most challenging part of some methods.

To get you started, **I have implemented one of the constructors and the toString() method for you. See the**

code and the comments in MyStringBuilder.java. These methods may help you to see how to approach the other methods in your class. Recitation Exercise 4 will also be very helpful – it is a different class and a singly-linked list, but once you understand how the copy constructor and equals() method for the LList<T> class work, you should better be able to implement some of the MyStringBuilder methods. I will post the solution for Recitation Exercise 4 after all of the sections have completed their meetings.

After you have completed your MyStringBuilder class, you will test it in two ways:

1) You will compile and run the program **Assig2.java**. This program is provided to you and **must be used as is.** The idea is similar to that of the test files for Assignment 1 – the program is a simple driver that tests most of the functionality of the MyStringBuilder class. Assig2.java will definitely test special cases of your operations, so be sure that these are handled. The output to this program is shown in file A2Out.txt. Your output should be **identical** to the data in this file.
2) You will compile and run the program Assig2B.java. This program is also provided to you and demonstrates the efficiencies of some of the MyStringBuilder operations. The overall run-times when run with your MyStringBuilder class **will depend a lot of the computer / system in which the program is run**, but the trend demonstrated should be similar to that of the sample output shown in file A2BOut.txt. **Note that this program may take a few minutes to run.**

Make sure you **submit ALL** of the following to get full credit:

1) All of the files that you wrote, well-formatted and reasonably commented:
   a) MyStringBuilder.java
   b) Any other Java files that you may have also written
2) The test driver programs provided to you, unchanged from how you downloaded them:
   a) Assig2.java
   b) Assig2B.java
3) Your completed Assignment Information Sheet. Be sure to help the TA with grading by clearly indicating here what you did and did not get working for the project.

The above files should be zipped up into a **single zip file** that you will submit for the assignment. As with Assignment 1, the idea from your submission is that your TA can compile and run your programs WITHOUT ANY additional files, so be sure to test them thoroughly **on the command line** before submitting them. If you cannot get the programs working as given, clearly indicate any changes you made and clearly indicate why (ex: "I could not get the indexOf() method to work, so I eliminated code that tested it") on your Assignment Information Sheet [one test the TA may do is to compile your MyStringBuilder.java file with a different copy of Assig2.java from the one you submitted – so if you modified Assig2.java in any way in order to get your program to work, it is essential that you state as much in your Assignment Information Sheet].

Extra Credit Ideas: If you want to get some extra credit, here are two ideas:

1) Implement some additional, non-trivial methods in your MyStringBuilder class. Look up StringBuilder in the Java API, choose one or more interesting methods that are not already required, and add them to your class. Alternatively, you could make up an interesting / useful method that is not already in the StringBuilder API. If you do this option, do not change the Assig2.java driver program. Rather, write a second driver to demonstrate your extra credit methods.
2) Add some additional test cases to Assig2B.java to see how MyStringBuilder compares to the standard Java StringBuilder class. Think about the benefits and liabilities of both the array-based StringBuilder and the linked-list based MyStringBuilder and empirically demonstrate them with some test cases.