Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Assignment 1 (4% - 40 points)

CSI2110/CSI2510 (Fall 2023)

Due: Monday October 9, 11:59PM

Late assignment policy: ***1min-24hs late are accepted with 30% off; no assignments accepted after 24hs late.***

Question 1. **[12 points =4+4+4]**

Decide if each of the following statements is true or false and give proof. For a true statement you need to identify the values for the constants c and n_0 as used in the definitions of big-O, *Ω and Θ and show the corresponding inequalities. For a false statement, you need to justify/prove why finding those constants is impossible.*

a) $n \log_2(n^2) + \sqrt{n}$ *is* $O(n \log_{10} n)$
b) $3^{n-1} + n^3$ *is* $\Omega(2^n)$
c) $n^3 \log_{10} n + 10\, n + 100$ *is* $\Theta(n^4)$

Question 2. **[8 points = 4+4]** *Consider the following algorithm given in Java code:*

```java
boolean orderedPrefixes(String s) {
// Input: an string S of length n with letters
// Output: print all sorted prefixes of S and return true if S is sorted
     boolean valid=true;
     int n=s.length();
     System.out.println(s.charAt(0));
     for (int i = 1; i < n;  i++) {
          if (s.charAt(i-1)> s.charAt(i)) {
               valid = false;
               break;
              }
          else {
               for (int j=0; j<= i; j++)
                    System.out.print(s.charAt(j));
               System.out.println();
               }
          }
      return valid;
      }
```

Note: *when giving the big-Oh , give the tightest upper bound possible. For example, if you can prove that* f(n) *is* O(n), *and that* f(n) *is* O(n^2), *choose the tighter upper bound, i.e.* f(n) *is* O(n).

(a) *(4 pts) Give a big-Oh for* T(n), *the **worst-case** running time of this algorithm for an input string of length* n. *Explain how you obtained this worst case.*

(b) *(4 pts) Give a big-Oh for* B(n), *the **best-case** running time of this algorithm for an input string of length* n. *Explain the type of inputs (sample inputs) that will give this best case.*

Question 3. **[8 points]**

(a) *Suppose you have a deque D containing the numbers (1,2,3,4,5,6,7,8) in this order. Suppose further that you have an initially empty queue Q. Give a code fragment that uses only D and Q (and no other variables), and results in D storing the elements in the order (1,2,3,5,4,6,7,8).*

(b) *Repeat the previous problem using the deque D and an initially empty stack S.*

Question 4. **Cool numbers [12 points]**

A number is COOL if the sum of two "opposite" digits in base 10 is always 10.
More precisely if the digits are a_d a_{d-1} ... a_1 a_0 then the number is cool if
$a_d+a_0=10$, $a_{d-1}+a_1=10$, ..., $a_1+a_{d-1}=10$, $a_0+a_d=10$.

For example, 1829 is cool, 18529 is cool, 721933 is not cool.
Note that 7212983 is not cool since a_3=2 and $2+2\neq 10$, but 7215983 is cool since a_3=5 and 5+5=10.

In this problem, the digits of the number are stored in nodes of a singly-linked list L. You are only allowed to read or process each list node only once.
Give pseudocode or Java-like code for a method that uses a stack and a queue to determine if the number is cool.

```
boolean isCool( LinkedList<Integer> L) {
// Write a code that returns true if the number is cool and false if
it is not.

}
```

Note: You do not need to give the implementation for the stack or the queue. Just assume these classes exist and have their standard methods.
In addition to the queue and stack, you are only allowed a constant amount of temporary variables, so you cannot use another array, for example.