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 (3% - 12 points)

CSI2110/CSI2510 (Fall 2021)

Due: Thursday Sep 23, 11:59PM

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

Question 1. **[6 points]**

Decide if each of the following statements is true or false and give a 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. For a false statement you need to justify why finding those constants is not possible.

a) $\log_{10}(n^2)$ is O($\log_2 n$)

b) $n \times (10\, n^2 - 2\sqrt{n}\,)$ is $\Omega(n^3)$

c) $(n \sin n)^2 + 100$ is $\Theta(n^2)$

Question 2. **[6 points]**

Suppose the only edit that can be done on strings is to replace 1 character. Given two strings having the same length `n`, the following function checks if they are one edit (or zero edit) away.

Examples:
marley, barley -> true
chip, chin -> true
lex, lox -> true
mule, maze -> false
aabb,abba -> false
abcdef, xbyzvw -> false

```
boolean oneEdit(String s1, String s2) {
// precondition s1.length() == s2.length()
  boolean foundDifference=false;
  n = s1.length();
  for (i=0; i< n; i++) {
      if (s1.charAt(i) != s2.charAt(i)){
          if (foundDifference) {
              return false;
          }
          foundDifference=true;
      }
  }
  return true;
}
```

(a) (4 points) Give the time complexity (also called running time) of this algorithm as a function of *n* using the big-O notation, for both the **worst case** and the **best case**. Give the simplest possible expression inside each big-Oh and the smallest possible function inside the big-O. Justify how you obtained the worst case and best case time complexities.

(b) (2 points) Now, the definition of one edit has been modified, so that any number between 0 and 100 replacements can be done in a single edit. Modify the above function to detect whether no more than 100 replacements have been done on the string; continue assuming both strings have the same length *n*. Your algorithm must have the same **big-Os** as the given algorithm for the **worst case** and for the **best case** running times**;** justify why this is the case.
Note: All pairs of strings in the previous example would have the new algorithm return true. An example that returns false would have to have at least 101 unmatching characters.