Université d'Ottawa
Faculté de génie

École d'ingénierie et de technologie de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information Technology and Engineering

Assignment 1 (5%, 100 marks)

CSI2110 – Fall 2017

Please submit by 23:55, October 2, 2017 via virtual campus as a single PDF file; late assignments are accepted up to 24hs late with 30% penalty.

Question 1. **[30 points; 5pt each]**

Using definitions and theorems seen in class, determine if each of the following functions is *$O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$ (*3 answers per item). Justify each answer with a proof. (*Note: all logs in this question are in base 2)*

a) $(n+1)(n+8)$
b) $n^2+\log n$
c) $(n+8) \log n$
d) $10n^3$
e) $\log(n^{10}+n^2)$
f) $4n+5$

Question 2. **[40 points]**

Consider an array **Age** with **n** elements representing the ages of individuals within a group in arbitrary order.

The following algorithm (given in Java-like pseudocode) outputs indexes of two individuals that are the closest in age among pairs of individuals whose ages are stored in the array. Note that if there is more than one pair of individuals with the same age difference we simply return one such pair. In the code below, abs(x) computes the absolute value of x.

For example, if Age[0]=25; Age[1]=50; Age[2]=15; Age[3]=52, the algorithm will output 1 and 3 since individual 1 and 3 have age difference equals to 2, while any other pair of individuals have larger age difference.

```
1.  x=0; y=1;
2.  int minDifference = abs(Age[0]-Age[1]);
3.  for (i=0; i<n-1; i++)
4.     for (j=i+1; j<n; j++) {
5.                int difference = abs(Age[i]-Age[j]);
6.                if (difference < minDifference) {
7.                      minDifference=difference;
8.                      x=i;
9.                      y=j;
10.                }
11. return x, y;
```

A) (10 pt) Give the worst-case number of comparisons of array elements that this algorithm performs (how many executions of line 6) exactly, as a function of n.
B) (10 pt) Give a big-Oh estimate for the number of comparisons calculated in the previous part. Justify your answer.
C) (14 pt) Now, suppose the input array **Age** is given in sorted (increasing) order. Give a linear time algorithm that solves the same problem, i.e. your algorithm must perform number of comparisons that is O(n).
D) (7 pt) Briefly justify why the number of comparisons is O(n).

Question 3. **[30 points]**

Someone proposes an algorithm to find the median of a series of N numbers (all different and given in a random order) contained in a simple array, where N is an odd number.

For example, if N=5 and the numbers in the array are 13,2,7,4,3 the median is 4 (the "middle" element).

The idea of this programmer is to go over each number and to find how many numbers in the array are smaller than this one. You do this for every number until you find one for which there are exactly (N-1)/2 smaller numbers; at this point you may stop and this number is the median.

A) (15 pt) Give a pseudo-code description of the algorithm described above.

B) (15 pt) Give a big-Oh complexity for the best-case and for the worst-case running time (of course, give the tightest big-Oh you can for each case). Justify your answers.