CS 2210a Data Structures and Algorithms
Assignment 1 (20 marks)
Due September 26

Put your assignment in a letter-size envelope labelled with your name and course number and drop it in the CS2210 locker (the locker is on the third floor of the Middlesex College Building, beside room MC300) by 11:59 pm on the due date. You need to print and fill out an assignment submission form

http://www.csd.uwo.ca/courses/CS2210a/submForm.html.

Put the submission form in the envelope along with your assignment.

When you are asked to find the time complexity of an algorithm you are required to give a big-Oh characterisation in terms of n of the running time of the algorithm. You might find the following fact useful: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

1. (3 marks) Use the definition of "big Oh" to prove that $\frac{2}{n} + 3$ is $O(1)$.

2. (3 marks) Use the definition of "big Oh" to prove that $3n$ is not $O(\sqrt{n})$.

3. (3 marks) Let $e(n)$, $f(n)$, $g(n)$, and $h(n)$ be non-negative functions. Assume that $f(n)$ is $O(g(n))$ and that $e(n)$ is $O(h(n))$. Use the definition of "big Oh" to show that $f(n) + e(n)$ is $O(g(n) + h(n))$.

4. Let A be an array storing n integer values.

   i. (4 marks) Write in pseudocode an algorithm that, without sorting the array, finds whether there is at least one pair of values $A[i]$, $A[j]$ such that $A[j] = A[i] + 1$ **and** $j > i$. If such a pair exists the algorithm must return the value `true`, otherwise it must return the value `false`. For example if A is the following array:

9	7	4	6	9	1	7	5	3	7
0	1	2	3	4	5	6	7	8	9

   Then, the algorithm should return the value `true` as $A[6] = A[3] + 1$ (and also $A[7] = A[2] + 1$). However, if the array A is as follows:

4	7	3	9	1	1	7	3	9	6
0	1	2	3	4	5	6	7	8	9

   Then, the algorithm must return the value `false`.

   ii. (4 marks) Explain what the worst case for the algorithm is and compute the time complexity of the algorithm in the worst case. You must explain how you computed the time complexity.

5. (3 marks) Consider the following algorithm:

```
**Algorithm** Foo ($n$)
    $x \leftarrow 0$
    **for** $i \leftarrow 0$ **to** $n \times n$ **do**
        **for** $j \leftarrow 0$ **to** $i$ **do**
            $x \leftarrow x + i$
    **return** $x$
```

Compute the time complexity of this algorithm in the worst case. You must give the order of the time complexity using "big-Oh" notation and you must explain how you computed the time complexity.

6. (2 marks) **Optional question**. Download from the course's website the java class `Search.java`, which contains implementations of 3 different algorithms for solving the search problem:

   - `LinearSearch`, of time complexity $O(n)$.
   - `QuadraticSeach`, of time complexity $O(n^2)$.
   - `FactorialSearch`, of time complexity $O(n!)$.

   Modify the `main` method so that it computes the worst case running times of the above algorithms for the following input sizes:

   - `FactorialSearch`, for input sizes $n = 5, 8, 9, 10, 11$.
   - `QuadraticSeach`, for input sizes $n = 5, 10, 100, 1000, 2000$.
   - `LinearSearch` for, input sizes $n = 5, 10, 100, 1000, 2000, 10000$.

   Print a table indicating the running times of the algorithms for the above input sizes. You do not need to include the code for the Search class.