CS 2210a Data Structures and Algorithms
Assignment 1 (20 marks)
Due September 24

Put your assignment in a **letter size** envelope labelled with your name and course number and drop it in the CS2210 locker (locker 300 located on the third floor of the Middlesex College Building, beside room MC300) by 11:59 pm on September 24. 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.

Remember that 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 facts useful: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

1. (3 marks) Use the definition of "big Oh" to prove that $n^2 + 100n$ is $O(n^2)$.

2. (3 marks) Use the definition of "big Oh" to prove that $n^3 + 2n^2$ is not $O(n^2)$.

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 $O(h(n))$. Show that $f(n) + e(n)$ is $O(g(n) + h(n))$.

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

   - (4 marks) Write an algorithm that given such an array A and an integer value k it returns the value *true* if there are two different integers in A that sum to k, and it returns *false* otherwise. For example, if A is the following array

4	7	3	9	-1	2	11
0	1	2	3	4	5	6

     and $k = 10$, then the algorithm returns the value *true* as, for example, $A[1] + A[2] = 10$ (note that also $A[4] + A[6] = 10$). For the same array A above and $k = 17$, the algorithm must return the value *false* as no two values in A add to 17.

   - Prove that your algorithm is correct:

     (a) (1 mark) Show that the algorithm terminates.

     (b) (2 marks) Show that the algorithm always produces the correct answer.

   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 give the order of the time complexity using "big-Oh" notation and you must explain how you computed the time complexity.

5. (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 $n = 5, 8, 9, 10, 11$. If you dare, run the algorithm for $n = 12$.
- `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 your code for the Search class.