CS 2210a Data Structures and Algorithms
Assignment 1 (20 marks)
Due October 1 at 11:59 pm.

Put your assignment in a 9"×12" 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 October 1. 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 characterization in terms of n of the running time of the algorithm.

1. (3 marks) Use the definition of "big Oh" to prove that $\frac{1}{n}$ is $O(1)$.
2. (3 marks) Use the definition of "big Oh" to prove that n is not $O(\sqrt{n})$.
3. (3 marks) Let $f(n)$ and $g(n)$ be non-negative functions such that $f(n)$ is $O(g(n))$. Use the definition of "big Oh" to prove that $k(f(n) + g(n))$ is $O(g(n))$ where k is a positive constant.
4. Let A be an array storing n distinct integer values. We say that A is *balanced* if for every value $A[i]$ there is another value $A[j]$, $i \neq j$, $0 \leq i, j \leq n-1$, such that $A[i] + A[j] = 0$. For example, the following array A is balanced as $A[0] + A[4] = A[1] + A[3] = A[2] + A[5] = 0$. However, the following array B is not balanced as there is no value $B[i]$ such that $B[1] + B[i] = 0$.

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

A

-2	4	-7	2	7	-3
0	1	2	3	4	5

B

   - (4 marks) Write pseudocode for an algorithm that given an array A of size n it returns the value *true* if A is balanced and it returns *false* otherwise.
   - 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:
   http://www.csd.uwo.ca/Courses/CS2210a/
   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.