CS 2210a Data Structures and Algorithms
Assignment 1 (20 marks)
Due September 29 at 11:59 pm.

You need to print and fill out an assignment submission form. The form can be downloaded from *http://www.csd.uwo.ca/courses/CS2210a/submForm.pdf*.
You must staple the submission form at the front of your assignment, so the submission form is the cover page. Drop your assignment in the CS2210 locker (located on the third floor of the Middlesex College Building, beside room MC300) by 11:59 pm on September 29.

For questions 1 and 3 proceed as follows:

1. First explain what needs to be proven: "We need to find constants $c > 0$ and $n_0 \geq 1$ integer such that ...".
2. For question 3 use the definition of "big Oh" to explain what it means for $f(n)$ to be $O(g(n))$ and for $g(n)$ to be $O(h(n))$.
3. Simplify the above inequalities.
4. Determine the values for c and n_0.

For question 2, if you use a proof by contradiction:

- First give the claim that you will assume false and from which you will derive a contradiction.
- Perform steps 1 and 3 as above
- Derive a contradiction.

---

1. (3 marks) Use the definition of "big Oh" to prove that $n^3 + 2n^2$ is $O(n^3)$.
2. (3 marks) Use the definition of "big Oh" to prove that $n(n+1)$ is not $O(n)$.
3. (3 marks) Let $f(n)$, $g(n)$, and $h(n)$ be non-negative functions such that $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$. Use the definition of "big Oh" to prove that $f(n)$ is $O(h(n))$.
4. Let A be an array storing n integer values. The goal is to design an algorithm that returns *true* if every value stored in A appears at least twice in the array; the algorithm returns *false* otherwise, i.e. if there is at least one value that appears only once in A.

   For example, for the following array A the algorithm must return *true* as values 1 and 6 appear twice and value 3 appears three times; however for array B it must return *false* as the value 3 appears only once.

3	6	1	6	3	1	3
0	1	2	3	4	5	6

A

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

B

   - (4 marks) Write pseudocode for an algorithm as described above.
   - 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.
   - (1 mark) Explain what the worst case for the algorithm is.

- (3 marks) 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.