CS 2210a Data Structures and Algorithms
Assignment 1 (20 marks)
Due September 27 at 11:59 pm.

Please submit on OWL a pdf file with your solution to the assignment. You are encouraged to type your answers to make it easier for the TA's to mark your work. If you decide to provide hand-written answers, please scan your solution and submit a pdf file through OWL.
Remember that concept assignments must be submitted by the due date; **no late concept assignments will be accepted**.

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 $O(f(n))$.
3. Simplify the above inequalities.
4. Determine values for c and n_0 that make the inequalities true.

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

- First give the claim that you will assume true and from which you will derive a contradiction.
- Use the definition of order to write the inequality from which you will derive the contradiction.
- Simplify the inequality and explain how you derive a contradiction from it.

---

1. (3 marks) Use the definition of "big Oh" to prove that $3n^3$ is $O(n^4)$.
2. (3 marks) Use the definition of "big Oh" to prove that $n^3 + n^2$ is not $O(n^2)$.
3. (3 marks) Let $f(n)$ and $g(n)$ be non-negative functions such that $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$. Use the definition of "big Oh" to prove that $f(n) - g(n)$ is $O(f(n))$.
4. Let A and B be two arrays, each storing n different integer values. The goal is to design an algorithm that returns *true* if A and B have no common values, and it returns *false* otherwise.

   For example, if the algorithm receives as input below arrays A and B the algorithm must return *true* as no value in A is also in B; however if the algorithm receives as input arrays A' and B it must return *false* as the values 3 and 4 in A' also appear in B.

6	8	1	7	5
0	1	2	3	4

   A

3	9	4	0	2
0	1	2	3	4

   B

8	3	1	4	6
0	1	2	3	4

   A'

   i. (4 marks) Write pseudocode for an algorithm as described above. You **cannot** use a hashmap or any other additional data structures. You **cannot** sort the arrays.

   ii. 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.

   iii. (1 mark) Explain what the worst case for the algorithm is.

    iv. (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 prints the worst case running times of the above algorithms for the following input sizes:

   - `FactorialSearch`, for input sizes $n = 7, 8, 9, 10, 11, 12$.
   - `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.