Instructions about how to hand in your solutions will come later.

CPSC 221: Summer 2015
Theory Assignment 1

History of Non-Trivial Changes:
- May 22, 2015 – Posted
- May 22, 2015 – Error in problem 3 corrected: $O(n^3)$ changed to $\Omega(n^3)$

Due Date: No later than Wednesday, May 27th at 11:59pm.
Late policy: Late submissions will receive no marks.

Note, you *may* work with a partner on this assignment. In order to do so, you must turn in **one assignment only**, with *both partners'* identifying information clearly stated (as per question 0). You ***may not*** work in groups of three or more.

0. Make sure to clearly identify your name, student number, 4-character ugrad ID, and your lab section. If you are not sure what lab section you are in, now is the time to find out. If this information is missing it may delay the return of your assignment, and *will result in marks lost*!

1. Write a *recursive* algorithm (step-by-step descriptions or pseudocode, not C++ code, Java, or Racket) that returns the sum of every element of an array with a value less than x.

2. The Traveling Salesman problem (TSP) is a famous problem for which there is no known, tractable solution (though efficient, approximate solutions exist). Given a list of cities and the distances in between, the task is to find the shortest possible tour (a closed walk in which all edges are distinct) that visits each city exactly once.

Consider the following algorithm for solving the TSP:

```
n = number of cities
m = n x n matrix of distances between cities
min = (infinity)
for all possible tours, do:
        find the length of the tour
        if length < min
                min = length
                store tour
```

a) What is the complexity of this algorithm in terms of n (number of cities)? You may assume that matrix lookup is O(1), and that the body of the if-statement is also O(1). You need not count the if-statement or the for-loop guard (i.e., conditional) checks, etc., or any of the initializations at the start of the algorithm. Clearly show the justification for your answer.

b) Given your complexity analysis, assume that the time required for the algorithm to run when n=10 is 1 second. Calculate the time required for n=20 and show your work.

Instructions about how to hand in your solutions will come later.

3. Use the definition of big-Ω that we have discussed in class to show the following (be sure to provide appropriate values for c and n_0):

$$\sum_{k=3}^{n}(k^2 - 2k) \quad \text{is } \Omega(n^3)$$

4. Suppose an algorithm solves a problem of size x in at most the number of steps listed in each question below. Calculate the asymptotic time complexity (big-Θ or big-theta) for each example below. Show your work, including values for c and x_0 along the way.

a) $T(x) = 1$
b) $T(x) = 5x - 2$
c) $T(x) = 3x^3 + 2 + x^2$
d) $T(x) = log\ (x\ *\ 2x!)$