Comp 352 - Data Structures and Algorithms
Fall 2014 - Assignment 1

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

Due date and time: **Friday, September 26, 2014 - 11:59 PM**

Written Questions (50 marks):

Q.1 Consider the following code fragments (a) and (b) where *n* is the variable specifying data size and C is an integer constant with C ≥ 2. What is the big-O time complexity in terms of *n* in each case? Show all necessary steps.

(a)
```
sum ← 0
for i ← 1 to n do
    for j ← 1 to n do
        k ← 1
        while k < n do
            k ← k * C
            sum ← sum + 1
```

(b)
```
sum ← 0
for i ← 1 to n do
    for j ← 1 to i do
        k ← 1
        while k < n do
            k ← k + C
            sum ← sum + 1
```

Q.2a. What is the big-O (O(*n*)) and big-Omega (Ω(*n*)) time complexity for the following algorithm in terms of *n*? Show all necessary steps:

Algorithm MyAlgorithm (A)
```
Input: Array A storing n ≥ 1 integers.
Output: Possibly modified Array A
for I ← 0 to A.length - 1 do
    for j ← A.length downto i + 1 do
        if A[i] > A[j - 1]
            s ← A[j - 1]
            A[j - 1] ← A[i]
            A[i] ← s
return A
```

Q.2b. Document the hand-run of MyAlgorithm for array A = (4,8,5,1,2). What is the resulting A?
Q.2c. Describe the functionality of MyAlgorithm. What can be asserted about its result given an arbitrary integer array A as input?

Q.3. Prove or disprove the following statements (in order to do this properly you have to apply the definition of Big-O or its relatives)

a) $n \log n$ is $O(n^5)$
b) $30n^3 + 6000n^4 + 4000n^2 - 10$ is $\Theta(n^4)$
c) $10000000n^2 + 0.0000001n^4$ is $O(n^2)$
d) $1000000n^2 + 0.0000001n^3$ is $\Omega(n^3)$
e) $n!$ is $\Omega(2^n)$
f) n^n is $O(n!)$

Programming Questions (50 marks):

In class, we discussed about the two versions of Fibonacci number calculations: BinaryFib(n) and LinearFibonacci(n) (refer to your slides and the text book). The first algorithm has exponential time complexity, while the second one is linear.

a) In this programming assignment, you will design in pseudo code and implement in Java two versions of Tetranacci calculators and experimentally compare their runtime performances. Tetranacci numbers are a more general version of Fibonacci numbers and start with four predetermined terms, each term afterwards being the sum of the preceding four terms. The first few Tetranacci numbers are:

   0, 0, 0, 1, 1, 2, 4, 8, 15, 29, 56, 108, 208, 401, 773, 1490, …

For that, with each implemented version you will calculate Tetranacci(5), Tetranacci(10), etc. in increments of 5 up to Tetranacci (100) (or higher value if required for your timing measurement) and measure the corresponding run times (for instance, Tetranacci(10) returns 56). You can use Java's built-in time function for this purpose. You should redirect the output of each program to an *out.txt* file. You should write about your observations on timing measurements in a separate text file. You are required to submit the two fully commented Java source files, the compiled executables, and the text files.

b) Briefly explain why the first algorithm is of exponential complexity and the second one is linear (more specifically, how the second algorithm resolves some specific bottleneck(s) of the first algorithm). You can write your answer in a separate file and submit it together with the other submissions.

c) Do any of the previous two algorithms use tail recursion? Why or why not? Explain your answer. If your answer is "No" then

   (i) Design the pseudo code for a tail recursive version of Tetranacci calculator;
   (ii) Write the corresponding Java program and repeat the same experiments as in part (a) above.

   You will need to submit both the pseudo code and the Java program, together with your experimental results. Keep in mind that Java code is **not** pseudo code.

<u>See course outline for exact submission format.</u> Submit all your answers to written questions in PDF (no scans of handwriting) or text formats only. Please be concise and brief (less than ¼ of a page for each question) in your answers. For the Java programs, you must submit the source files together with the compiled executables. The solutions to all the questions should be zipped together into one .zip or .tar.gz file and submitted via EAS. You may upload at most one file to EAS.

For the programming component, only one submission is to be made per group (≤ 2 students). You must make sure that you upload the assignment to the correct directory of Assignment 1 using EAS. Assignments uploaded to the wrong directory will be discarded and no re-submission will be allowed.