Department Computer Science and Software Engineering
Concordia University
COMP 352: Data Structures and Algorithms
Fall 2016
Assignment 1
Due date and time: Friday September 30, 2016 by midnight

Written Questions (50 marks):

Question 1

a) Given an array of integers of any size, $n \geq 1$, write an algorithm as a pseudo code (not a program!) that would reverse the content of the left half of the array, while leaving the right half intact. If the given array is of an odd size, then the left half is considered to have *n/2 + 1* elements. Additionally, your algorithm **cannot** use any auxiliary storage such as array to perform what is needed.
b) What is the time complexity of your algorithm, in terms of Big-O?
c) What is the space complexity of your algorithm, in terms of Big-O?

Question 2

Prove or disprove the following statements, using the relationship among typical growth-rate functions seen in class.

a) $n^5 \log n$ is $O(n^7)$
b) $10^8 n^2 + 5n^4 + 5000000n^3 + n$ is $\Theta(n^4)$
c) n^n is $O(n!)$
d) $0.01n^2 + 0.0000001n^4$ is $\Theta(n^2)$
e) $1000000n^2 + 0.0000001n^5$ is $\Omega(n^3)$
f) $n!$ is $\Omega(2^n)$

Question 3

a) What is the big-O (*O(n)*) and big-Omega (*Ω(n)*) time complexity for algorithm *MyAlgorithm* below in terms of *n*? Show all necessary steps.
b) Trace (hand-run) *MyAlgorithm* for an array A = (6,12,7,3,5). What is the resulting A?
c) What does *MyAlgorithm* do? What can be asserted about its result given any arbitrary array A of n integers?
d) Can the runtime of *MyAlgorithm* be improved easily? how?
e) What type of recursion is used in *MyAlgorithm*? Is *MyAlgorithm* tail-recursive?

```
Algorithm MyAlgorithm(A, n)
  Input: Array of integer containing n elements
  Output: Possibly modified Array A
    done ← true
    j ← 0
    while j ≤ n - 2 do
      if A[j] > A[j + 1] then
        swap(A[j], A[j + 1])
        done:= false
      j ← j + 1
    end while
    j ← n - 1
    while j ≥ 1 do
      if A[j] < A[j - 1] then
        swap(A[j - 1], A[j])
        done:= false
      j ← j - 1
    end while
  if ¬ done
    MyAlgorithm(A, n)
  else
    return A
```

Programming Questions (50 marks):

In class, we discussed the two versions of recursive Fibonacci number calculations: BinaryFib(n) and LinearFibonacci(n) (refer to your slides and the text book).

In this programming assignment, you will design an algorithm (in pseudo code), and implement (in Java), two recursive versions of an *Oddonacci* calculator (similar to the ones of Fibonacci) and experimentally compare their runtime performances. Oddonacci numbers are inspired by Fibonacci numbers but start with three predetermined values, each value afterwards being the sum of the preceding three values. The first few Oddonacci numbers are:

1, 1, 1, 3, 5, 9, 17, 31, 57, 105, 193, 355, 653, 1201, 2209, 4063, 7473, 13745, 25281, 46499, ...

For that, with each implemented version you will calculate Oddonacci(5), Oddonacci (10), etc. in increments of 5 up to Oddonacci(100) (or higher value if required for your timing measurement) and measure the corresponding run times. For instance, Oddonacci(10) returns 105 as value. 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 the 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.

a) 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.

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

   Can a tail-recursive version of Oddonacci calculator be designed?

   i. If yes; write the corresponding pseudo code for that tail recursion algorithm and implement it in Java and repeat the same experiments as in part (a) above.
   ii. If no, explain clearly why such tail-recursive algorithm is infeasible.

   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.

The written part must be done individually (no groups are permitted). The programming part must be done in groups of two students. 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, you must make sure that you upload the assignment to the correct directory of Assignment 1 using EAS. Indicate clearly in your programming submission the group members (full names and student ids). Assignments uploaded to the wrong directory will be discarded and no resubmission will be allowed.