# COMP 3270 Assignment 1

## 100 points. Due at **beginning of class** Friday, May 31, 2013

Instructions:

1. This is an individual assignment. You should do your own work. Any evidence of copying will result in a zero grade and additional penalties/actions.
2. Submissions not handed to the instructor at the beginning of class (late arrivals not excused) on the due date **will not** be accepted unless prior permission has been granted or there is a valid and verifiable excuse.
3. Think carefully; formulate your answers, and then write them out concisely using English, logic, mathematics and pseudocode (no programming language syntax).
4. Type your final answers in this Word document. You must hand a **printed copy** to the instructor.
5. Don't turn in handwritten answers with scribbling, cross-outs, erasures, etc. If an answer is unreadable, it will earn zero points. **Neatly and cleanly handwritten submissions are also acceptable**.

1. (3 points) *Computational problem solving: Estimating problem solving time:* Suppose there are three algorithms to solve a problem- a O(n) algorithm (A1), a O(nlogn) algorithm (A2) and a $O(n^2)$ algorithm (A3) where log is to the base 2. Using the techniques and assumptions in slide set L2-Buffet(SelectionProblem).ppt, determine how long in seconds it will take for each algorithm to solve a problem of size 200 million. You must show your work to get credit, i.e., a correct answer without showing how it was arrived at will receive zero credit.

2. (6 points) *Computational problem solving: Problem specification*
Suppose you are asked to develop a mobile application to provide turn by turn directions on a smartphone to an AU parking lot in which there are at least five empty parking spots nearest to a campus building that a user selects. Assume that you can use the Google Map API for two functions (only) – display campus map on the phone so user can select a campus building, and produce turn-by-turn directions from a source location to a destination location – where any location in the map is specified as a pair (latitude, longitude). Also assume that there is an application called AUparking that you can query to determine the # of vacant spots in any parking lot specified as a pair (latitude, longitude). Specify the problem to a level of detail that would allow you to develop solution strategies and corresponding algorithms: State the problem specification in terms of (1) inputs, (2) data representation and (3) desired outputs; no need to discuss solution strategies.

3. (5 points) *Computational problem solving: Developing strategies*
Explain a correct and efficient **strategy** to check what the maximum difference is between any pair of numbers in an array containing n numbers. Your description should be such that the strategy is clear, but at the same time the description should be at the level of a strategy, not an algorithm. Then state the total number of number pairs any algorithm using the strategy "compute the difference between every number pair in the array and select that pair with the largest difference" will have to consider as a function of n.

4. (7 points) *Computational problem solving: Understanding an algorithm and its strategy by simulating it on a specific input:*
Understand the following algorithm. Simulate it mentally on the following four inputs, and state the outputs produced (value returned) in each case: (a) A: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; (b) A: [–1, –2, –3, –4, –5, –6, –7, –8, –9, –10], ; (c) A: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], (d) A: [–1, 2, –3, 4, –5, 6, 7, –8, 9, –10].

```
        Algorithm-1 (A:array[1..n] of integer)
        sum, max: integer
1       sum = 0
2       max = 0
3       for i = 1 to n
4               sum = 0
5               for j = i to n
6                       sum = sum + A[j]
7                       if sum > max then
8                               max = sum
9       return max
```

Output when input is array (a) above:

Output when input is array (b) above:

Output when input is array (c) above:

Output when input is array (d) above:

What does the algorithm return when the input array contains all negative integers?

What does the algorithm return when the input array contains all non-negative integers?

What does the algorithm return when the input array contains negative, zero and positive integers?

5. (9 points) *Computational problem solving: Calculating approximate complexity:*
Using the approach described in class (L5-Complexity.pptx), calculate the approximate complexity of Algorithm-1 above by filling in the table below.

| Step | Big-Oh complexity |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| Complexity of the algorithm | |

6. (18 points) Calculate the detailed complexity T(n) of Algorithm-1. Fill in the table below, then determine the expression for T(n) and simplify it to produce a polynomial in n.

| Step | Cost of each execution | Total # of times executed |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |

T(n) =

7. (5 points) *Computational problem solving: Proving correctness/incorrectness:*
Is the algorithm below correct or incorrect? Prove it! It is supposed to count the number of all identical integers that appear consecutively in a file of integers. E.g., if f contains 1 2 3 3 3 4 3 5 6 6 7 8 8 8 8 then the correct answer is 9

```
Count(f: input file)
count, i, j : integer          //local variables
count=0
while end-of-file(f)=false
        i=read-next-integer(f)
        if end-of-file(f)=false then
                j=read-next-integer(f)
                if i=j then count=count+1
return count
```

8. (10 points) *Computational problem solving: Proving correctness:* Complete the proof by contradiction this algorithm to compute the Fibonacci numbers is correct.

```
function fib(n)
1. if n=0 then return(1)
2. if n=1 then return(1)
3. last=1
4. current=1
5. for i=2 to n do
6.      temp=last+current
7.      last=current
8.      current=temp
9. return(current)
```

1. Assume the algorithm is incorrect.
2. Fibonacci numbers are defined as $F_0=1$, $F_1=1$, $F_i=F_{i-1}+F_{i-2}$ for $i>1$.
3. So the assumption in (1) implies that there is at least one input parameter $n=k$, $k\geq 0$, for which the algorithm will produce an incorrect answer.
4. _____
   _____
   So in both cases the algorithm returns the correct answer.
5. This implies that there has to be at least one integer $k>1$, so that when $n=k$ the algorithm does not return the correct answer $F_k=F_{k-1}+F_{k-2}$.
6. When $n=k$ and $k>1$ _____, and steps 3-9 will be executed.
7. If k=2, the for loop in steps 5-8 will be executed exactly once. By step 6, temp = last + current = $1 + 1 = F_0 + F_1$. Then step 7 updates last to be equal to current = $F_1$. Step 7 updates current to be equal to temp which is $F_0 + F_1$. So the value returned in step 9 is current = $F_0 +$

$F_1 = F_2$. This is the correct answer. So the k for which the algorithm fails must be greater than 2.

8. If k=3,

   _____
   _____
   _____

9. But if k= 4,

   _____
   _____
   _____

10. The above argument can be repeated to show that

    _____
    _____
    _____.

11. That is, for all k > 1 the algorithm returns the correct k-th Fibonacci number.
12. So there is no k for which the algorithm will return a value not equal to $F_{k-1}+F_{k-2}$. This contradicts (3).
13. Therefore, the algorithm must be correct.

9. (a) (6 points) *Computational problem solving: Algorithm design:* Describe a recursive algorithm to reverse a string that uses the strategy of swapping the first and last characters and recursively reversing the rest of the string. Assume the string is passed to the algorithm as an array A of characters, A[p…q], where the array has starting index p and ending index q, and the length of the string is n=q–p+1. The algorithm should have only one base case, when it gets an empty string. Assume you have a swap(A[i],A[j]) function available that will swap the characters in cells i and j. Write the algorithm using pseudocode without any programming language specific syntax. Your algorithm should be correct as per the technical definition of correctness.

(b) (8 points) Draw your algorithm's recursion tree on input string "i<33270!"- remember to show inputs and outputs of each recursive execution including the execution of any base cases.

10. (10 points) *Computational problem solving: Proving correctness:*
Function g (n: nonnegative integer)

if $n \leq 1$ then return(n)

else return(5*g(n-1) – 6*g(n-2))

Prove by induction that algorithm g is correct, if it is intended to compute the function $3^n-2^n$ for all $n \geq 0$.

Base Case Proof:

Inductive Hypothesis:

Inductive Step:

11. (13 points) *Computational problem solving: Proving correctness:* The algorithm of Q.8 can also be proven correct using the Loop Invariant method. The proof will first show that it will correctly compute $F_0$ & $F_1$ by virtue of lines 1 and 2, and then show that it will correctly compute $F_n$, n>1, using the LI technique on the for loop. For this latter part of the correctness proof, complete the Loop Invariant below by filing in the blanks. Then complete the three parts of the rest of the proof.

Loop Invariant:

Before any execution of the for loop of line 5 in which the loop variable $i=k$, $2 \leq k \leq n$, the variable last will contain _____ and the variable current will contain _____.

Initialization:

Maintenance:

Termination: