COMP 3270
Introduction to Algorithms
Homework 1

Due: June 3^{rd}, Monday by 11:59 PM (midnight)
Please submit as a PDF document using Canvas

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 on the due date and time will incur late penalty (i.e., 10 point deduction for each late hour) 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 a Word (or Latex) document and submit online as a **PDF through Canvas**.
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. 6pts) *Computational problem solving: Estimating problem solving time*
Suppose that 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 presented in slide set L2-Buffet(SelectionProblem), 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 is derived will receive zero credit.

(2. 4pts) *Computational problem solving: Problem specification*
Problem Definition: You are a member of a software engineering team, which is tasked to develop a mobile application for the SmartCity initiatives sponsored by an enterprise. The application is a transport sharing tool that will connect a driver with empty seats to people traveling to a specific target location. The driver's application will query a remote service to determine specific requests submitted by customers, where each request is defined by the position of the person making the ride sharing request. The location of a person is specified as pair (latitude, longitude). Given a list of requests as input and the computed shortest distances in terms of turn-by-turn directions between each pair of locations, the application generates the shortest possible route that visits each location exactly once (to pick up a customer) and return to the target location.

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) discrete structures/data representation, and (3) desired outputs. **No need to discuss solution strategies**.

(3. 8pts) *Computational problem solving: Developing strategies*
Given a set of n numbers, explain a *correct* and *efficient* **strategy** to find the i largest numbers in sorted order. Your description should be such that the strategy is clear, but at the same time the description should be at the level of strategy (e.g., sort the numbers and list the i largest – you should devise a different strategy than this obvious one). Then state the total number of steps an algorithm that implements the strategy will have to consider as a function of n.

(4. 12pts) *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 Mystery (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?

(5. 10pts) *Computational problem solving: Calculating approximate complexity*
Using the approach described in class (L5-Complexity), calculate the approximate complexity of *Mystery* algorithm 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. 13pts) Calculate the detailed complexity T(n) of algorithm *Mystery* 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. 5pts) *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
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. 5pts) *Computational problem solving: Proving Correctness*
Let A be an algorithm that finds the kth largest of n elements by a sequence of comparisons. **Prove by contradiction** that A collects enough information to determine which elements are greater than the kth largest and which elements are less than it. In other words, you can partition the set around the kth largest element without making more comparisons.

(9. 8pts) *Computational problem solving: Proving Correctness*

```
Function g(n: nonnegative integer)
  if n <= 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 $n \geq 0$

(10. 5pts) *Computational problem solving: Proving Correctness*
Complete the proof by loop invariant method to show that the following algorithm is correct.

```
Algorithm Max(A: Array[1..n] of integer)
begin
  max = A[1]
  for i=2 to n
    if A[i] > max then max = A[i]
  return max
end
```

Initialization:

Maintenance:

Termination:

(11. 10pts) *Computational problem solving: Proving Correctness*
Complete the proof by loop invariant method to show that the following algorithm is correct.

```
Algorithm Convert(n: positive integer)
output: b (an array of bits corresponding to the binary representation of n)
begin
  t = n
  k = 0
  while (t > 0)
    k = k + 1
    b[k] = t mod 2
    t = t div 2   (div refers to integer division)
end
```

Use the following **loop invariant**: If m is the integer represented by the binary array $b[1..k]$ then $n = t2^k + m$

Initialization:

Maintenance:

Termination:

(12. 14pts) *Computational problem solving: Algorithm Design*
(a. 10pts) 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) (4pts) 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.