COMPUTER SCIENCE
AND SOFTWARE ENGINEERING

AUBURN UNIVERSITY
SAMUEL GINN COLLEGE OF ENGINEERING

COMP 3270
Introduction to Algorithms
Homework 2

Due: June 11th, 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 uploaded 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 using a document editor (e.g., Word) 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. 10pts)

Compare the following pairs of functions in terms of order of magnitude. In each case, say whether $f(n) = \mathcal{O}(g(n))$, $f(n) = \Theta(g(n))$, and/or $f(n) = \Omega(g(n))$

	f(n)	*g(n)*
a.	$100n + logn$	$n + (logn)^2$
b.	$logn$	$log(n^2)$
c.	$\frac{n^2}{logn}$	$n(logn)^2$
d.	$n^{\frac{1}{2}}$	$(logn)^5$
e.	$n2^n$	3^n

(2. 10pts)

```
Algorithm Mystery(A: Array [i..j] of integer)
i & j are array starting and ending indexes
 begin
   if i=j then return A[i]
   else
     k=i+floor((j-i)/2)
     temp1= Mystery(A[i..k])
     temp2= Mystery(A[(k+1)..j]
     if temp1<temp2 then return temp1 else return temp2
  end
```

(a) (1 pts) What does the recursive algorithm above compute?

(b) (3 pts) Develop and state the two recurrence relations exactly (i.e., determine all constants) of this algorithm by following the steps outlined in L7-Chapter4.ppt. Determine the values of constant costs of steps using directions provided in L5-Complexity.ppt. Show details of your work if you want to get partial credit.

(c) (6 pts) Use the Recursion Tree Method to determine the precise mathematical expression T(n) for this algorithm. First, simplify the recurrences from part (b) by substituting the constant "c" for all constant terms. Drawing the recursion tree may help but you do not have to show the tree in your answer; instead, fill the table below. Use the examples worked out in class for guidance. Show details of your work if you want to get partial credit.

Level	Level number	Total # of recursive executions at this level	Input size to each recursive execution	Work done by each recursive execution, excluding the recursive calls	Total work done by the algorithm at this level
Root	0				
One level below root					
Two levels below root					
The level just above the base case level					
Base case level					

(d) (1 pts) Based on T(n) that you derived, state the order of complexity of this algorithm:

3. (10 pts) $T(n) = 7T(n/8) + cn; T(1) = c$. Determine the polynomial T(n) for the recursive algorithm characterized by these two recurrence relations, using the Recursion Tree Method. Drawing the recursion tree may help but you do not have to show the tree in your answer; instead, fill the table below. You will need to use the following results, where and b are constants and x<1:

$$a^{log_b^n} = n^{log_b^a}$$

$$\sum_{i=0}^{\infty} x^i = 1/1 - x$$

level	Level number	Total # of recursive executions at this level	Input size to each recursive execution	Work done by each recursive execution, excluding the recursive calls	Total work at this level
Root	0				
1 level below	1				
2 levels below	2				
The level just above the base case level					
Base case level					

T(n) =

4. (5 points) Use the substitution method to prove the guess that is indeed correct when $T(n)$ is defined by the following recurrence relations: $T(n) = 3T(n/3) + 5; T(1) = 5$. At the end of your proof state the value of constant c that is needed to make the proof work.

Statement of what you have to prove:

Base Case proof:

Inductive Hypotheses:

Inductive Step:

Value of c:

5. (6 points) Find a counterexample to the following claim:

$f(n)$=$\mathcal{O}(s(n))$ and $g(n)$=$\mathcal{O}(r(n))$ imply $f(n) - g(n)$ = $\mathcal{O}(s(n) - r(n))$

6. (16 points) Guess a plausible solution for the complexity of the recursive algorithm characterized by the recurrence relations $T(n) = T(n/2) + T(n/4) + T(n/8) + T(n/8) + n; T(1) = c$ using the Substitution Method. (1) Draw the recursion tree to three levels (levels 0, 1 and 2) showing (a) all recursive executions at each level, (b) the input size to each recursive execution, (c) work done by each recursive execution other than recursive calls, and (d) the total work done at each level. (2) Pictorially show the shape of the overall tree. (3) Estimate the depth of the tree at its shallowest part. (4) Estimate the depth of the tree at its deepest part. (5) Based on these estimates, come up with a reasonable guess as to the *Big-Oh* complexity order of this recursive algorithm.
Your answer must explicitly show every numbered part described above in order to get credit.

7. (10 points) Use the Substitution Method to prove that your guess for the previous problem is indeed correct.

Statement of what you have to prove:

Base Case proof:

Inductive Hypotheses:

Inductive Step:

8. (9 points) Use the Master Method to solve the following three recurrence relations and state the complexity orders of the corresponding recursive algorithms.

(a) $T(n) = 2T(99n/100) + 100n$

(b) $T(n) = 16T(n/2) + n^3 lgn$

(c) $T(n) = 16T(n/4) + n^2$

9. (10 points) Use Backward Substitution (10 points) and then Forward Substitution (10 points) to solve the recurrence relations $T(n) = 2T(n-1) + 1$; $T(0) = 1$. In each case, do the following: (1) Show at least three expansions so that the emerging pattern is evident. (2) Then write out $T(n)$ fully and simplify using equation (A.5) on Text p.1147. (3) Verify your solution by substituting it in the LHS and RHS of the recurrence relation and demonstrating that LHS=RHS. (4) Finally state the complexity order of T(n).

You must show your work for parts (1)-(3) to receive credit.

10. (5 points) Solve the following recurrence relation. Give an exact solution:

$$T(n) = T(n-1) + n/2$$

;

$$T(1) = 1$$

11. (5 points) Prove that T(n), which is defined by the recurrence relation

$$T(n) = 2T(\lfloor n/2 \rfloor) + 2nlog_2 n$$

$$T(2) = 4$$

satisfies $T(n) = O(nlog^2 n)$

12. (4 points). Problem 3.1-3 (p.53) – Explain why the statement “The running time of algorithm is at least $O(n^2)$," is meaningless.