CSE 310 Assignment #2
(Max. Points: 30)

Due on: Friday, Sep. 18, 2020 11:59pm Arizona time

General Instructions:

- This is an individual assignment, please do not collaborate. If there's programming part, make sure that you write every line of your own code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report sent to the Dean's office.
- For all written exercises: **your answer should be clearly typed or written and must be saved in .pdf format**. Note: **unreadable answer receives no credits!**
- *For this assignment, you will need to submit your typed solution through the link posted on Canvas*, we do NOT accept any hand-in submissions or submissions sent through emails!
- Submission link will be closed automatically once the due date/time is past and **no late assignment will be accepted**.
- You will be allowed multiple times to submit the assignment before the due date/time, but we will only grade your last submission.

Objectives

- Asymptotic notation.
- Binary Search, MergeSort, QuickSort, recurrence, Master method, etc.
- Be familiar with OpenMP

Questions

1. [6 pts, 2 pts each]For each group of functions, sort the functions in increasing order of asymptotic (big-*O*) complexity.

A) **Group A**

$f_1(n) = n^{0.9999} logn$
$f_2(n) = n^2$
$f_3(n) = 1.00001^n$
$f_4(n) = n^{1.0001}$

B) **Group B**

$f_1(n) = 2^{100n}$
$f_2(n) = n\sqrt{n}$
$f_3(n) = 2^n$
$f_4(n) = 2^{2^{100n}}$

C) **Group C**

$f_1(n) = n^{\sqrt{n}}$
$f_2(n) = n^{10} \cdot 2^{n/2}$
$f_3(n) = n \cdot 2^n$
$f_4(n) = n!$

2. [3 pt] Solve the following recursion exactly by drawing a ***recursion tree***. (You will need to give an exact explicit solution for *T*).

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + n^2$$

3. [2 pts each, total 6 pts] Use the master method to give tight asymptotic bounds (θ bound) for the following recurrences. Specify clearly *a*, *b*, *f(n)* or ε value and which case you applied.

A) $T(n) = 2T\left(\frac{n}{3}\right) + 1$

B) $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

C) $T(n) = 3T\left(\frac{n}{5}\right) + n$

4. Given a one-dimensional array $A[1..n]$, we define a "bump" as a position where the element at this position is greater or equal to its adjacent element(s). For the following example, position 2 and 8 are such two bumps.

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
6	**7**	4	1	2	3	4	**5**
	↑						↑

Question #1[1 pt]: Does the bump always exist?

Question #2 [2 pts]: Design a linear algorithm (running time $O(n)$) to find such a bump if it exists. (Note: you just need to find one such bump, no need to find all the bumps if they exist). Clearly write your algorithm in pseudo-code similar as those algorithms in your textbook).

Question #3 [3 pts]: Can you do better? Consider a strategy similar as we used in binary search, design an algorithm which runs in $O(lgn)$ time for this problem, write your algorithm in pseudo-code and give the recurrence equation, solve it.

5. [3 pts] Illustrate the operation of MAX-HEAPIFY(*A*, 2) on the array A = { 15, 5, 2, 26, 11, 13, 8, 6, 9, 12, 3, 7} by re-drawing the tree for every swap.

6. [Total 6 pts] Read the notes on OpenMP (file *OpenMP.pdf*) and compile/execute the sample programs included in the zip file (file *openmp.zip*) as you go to familiarize yourself with the various OpenMP directives. Answer the following questions; it is expected that you will run each program multiple times. You may need to modify the program(s) to answer the questions. Be sure to explain the experiments you conducted.

1) [2 pts] For the program in the file *parfor.cc* vary the value of the variable *n* and the number of threads specified in the num threads clause. How are the iterations distributed among threads? Be sure to try out fewer iterations than threads, and more iterations than threads, etc. Write your observation.

2) [2 pts] For the program in the file *private.cc* explore the values of the variables *a* and *i* before, after, and inside the parallel region. Try initializing the variables before the *#pragma* and also not initializing them. Write your observation.

3) [2 pts] For the program in the file *reduction.cc* explore the run time of the reduction clause by varying the number of threads in the num threads clause Write your observation.