CSCD320 Homework2, Winter 2024, Eastern Washington University, Spokane, Washington.

Name: **EWU ID:**

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own solutions, without referring to anybody else's solution.

2. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.

3. Submission must be computer typeset in the **PDF** format and sent to the Canvas system. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this homework as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and are also used by nearly all the computer science and math professionals for paper writing.

4. Your submission PDF file must be named as: **firstname_lastname_EWUID_cscd320_hw2.pdf**
   (1) We use the underline '_' not the dash '-'.
   (2) All letters are in the lower case including your name and the filename's extend.
   (3) If you have middle name(s), you don't have to put them into the submission's filename.

5. Sharing any content of this homework and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Problem 1 (20 points)**.** *Suppose you are given an array A with n entries, with each entry holding a distinct number. You are told the sequence of values $A[1], A[2], \ldots, A[n]$ is* **unimodal***: For some index p between 1 and n, the values in the array entries increase up to position p in A and then decrease the remainder of the way until position n. That is, $A[1] < A[2] < \ldots < A[p]$ and $A[p] > A[p+1] > \ldots > A[n]$, for some index $p \in \{1, 2, \ldots, n\}$.* **Your task:** *find the "peak entry" p without having to read the entire array—in fact, by reading as few entries of A as possible. Show how to find the entry p by reading $O(\log n)$ entries of A. Describe your algorithmic idea, show its pseudocode, and explain why your algorithm's time complexity is $O(\log n)$, where n is the input array size.*

Problem 2 (20 points)**.** *Prove $T(n) = 2T(n/4) + n = O(n)$ using the inductive proof technique. That is, prove: there exist some positive constant n_0 and c, such that $T(n) \leq cn$ for all $n \geq n_0$.* (Hint: any positive values for n_0 and c are good as long as they can make the proof work.)

Problem 3 (20 points)**.** *Prove $T(n) = 2T(n/2) + n^2 = \Theta(n^2)$ using the recursion tree technique.*

Problem 4 (20 points)**.** *Solve the following recurrences using the* Master Theorem*. (You can directly give the results.)*

1. $T(n) = 8T(n/2) + 3n^2 - 9n$

2. $T(n) = 8T(n/2) + 2n^3 - 100n^2$

3. $T(n) = 4T(n/2) + n^2 + 5\log n$

4. $T(n) = 8T(n/2) + n^3 + n\log n$

5. $T(n) = 8T(n/2) + 4n$
6. $T(n) = 4T(n/2) + 2^{-10}n^4 - 6n^3$

Problem 5 (20 points). *Propose TWO example recurrences that CANNOT be solved by the* Master Theorem. *Note that your examples must follow the shape that* $T(n) = aT(n/b) + f(n)$*, where* n *are natural numbers,* $a \geq 1$*,* $b > 1$*, and* $f(n)$ *is an increasing and non-negative function. Explain why your recurrences cannot be solved by the master theorem.*