CSCD320 Homework4, Eastern Washington University, Spokane, Washington.

Name: **EWU ID:**

Please follow these rules carefully:

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 Blackboard system). Send in whatever you have had by the deadline.

3. Submission must be computer typeset in the **PDF** format and sent to the Blackboard 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_hw4.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. You are NOT allowed to share this homework in any way with others who are not in this class.

Problem 1 (20 points)**.** *Let $A[1 \ldots n]$ be a max-heap. The operation* `max-heap-delete(A,i)` *deletes the heap element $A[i]$ from the heap, so that the rest after the deletion is still a max-heap. Please gives the pseudocode of* `max-heap-delete(A,i)` *and its time complexity in big-oh notation. Hint: it's a bit more complicated that inserting a new key, and you can use the subroutines in the books if they are useful.*

Problem 2 (20 points)**.** *Give an $O(n \log k)$-time algorithm that uses a min-heap of size no more than k to merge k SORTED lists into one sorted list, where n is the total number of elements in all input lists. Give your algorithmic idea and pseudocode and explain why your algorithm has a time cost of $O(n \log k)$. Explain how you will use your algorithm for* `mergesort` *if the* `mergesort` *has a setting of k-way splitting.*

Problem 3 (20 points)**.** *Planning EWU's thanksgiving party: Everyone (including all the faculty, staff, and students) at EWU has one and only one direct supervisor except the president, so the supervisorship among the people of EWU can be represented as a rooted tree where a node is the direct supervisor of its child nodes (if the node has child nodes) and the president is the root of this tree.*

Now the HR office of EWU is planning the Thanksgiving dinner for the EWU community. In order to let everyone relax and have fun, all the people invited for the dinner should not have their direct supervisors invited. Now you, as the direct of EWU HR, are given the supervisorship tree and need to decide who should be invited. Your goal is to invite as many people as you can, as long as everyone can relax. Describe your strategy for picking the guests and explain why your strategy works. [Hint: It seems that you all should be invited and I should not, assuming I am your direct supervisor and you all do not have supervisees.]

Problem 4 (20 points)**.** *You are given a set of n* **unit-length** *tasks, i.e., each task has length one. However, each task i, for $1 \leq i \leq n$, can start at the earliest possible time r_i and must finish by its deadline d_i. For example, if $r_i = 3$ and $d_i = 8$, that means you can choose to start the ith task at any of these clock times:*

$3, 4, 5, 6, 7$, *because: 1) it is allowed to be started as early as at clock time* 3*, and 2) you must get it started by no later than clock time* 7 *because the task's deadline is* 8 *and its length is* 1*.*

You are asked to schedule a maximum number of tasks within the given constraint, so that all the picked tasks can be scheduled without conflicts. Describe your strategy on picking what tasks and how to schedule them. Explain why your strategy works.

Problem 5 (20 points)**.** *Suppose you are given a text of* 375 *characters drawn from the alphabet* {`a, b, c, d, e, f, g, h`} *and the frequency of each letter is:*

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
40	*30*	*90*	*20*	*60*	*120*	*10*	*5*

1. *Create a Huffman tree for this text (you may have multiple different Huffman tree for this text, but anyone is fine).*
2. *Show the Huffman code of each letter.*
3. *Compute the size of the Huffman code compressed version of this text in bits.*
4. *Calculate the compression ratio: compressed text size in bits / raw text size in bits, if you use 8-bit ASCII code for storing the raw text.*