University of Waterloo

CS240 Fall 2021 Midterm Instructions

***Availability Starts: Wednesday, October 27, 2021, 10:00AM** (Waterloo, Ontario time)
Availability Ends: Thursday, October 28, 2021, 10:00AM (Waterloo, Ontario time)
Allowed Aids: only course provided materials

*** Once you have downloaded the midterm** (see availability above) from (https://www.student.cs.uwaterloo.ca/~cs240/protect/index.php)**, your time has started** (see https://student.cs.uwaterloo.ca/~cs240/cgi-bin/displayExamTime.cgi for timer display)**.** You are allocated 2 hours to write the exam and an additional 1.5 hours to create one file with each question starting on a new page or individual files for each question and submit it/them to ***Crowdmark***. All work must be submitted within 3.5 hours of when you downloaded the exam or Thursday, October 28, 2021, 10:00AM, Waterloo, Ontario time, whichever comes ***first***. We will only mark work that has been submitted **to the proper question** on time.

Piazza: Public posts to Piazza will be turned off during the exam. *Only private posts* to instructors will be allowed in the case where you think you have found an error or lack of clarity (read the question again carefully first). We will put any errors or clarifications on Piazza in an ***Official Midterm Exam Clarifications*** post. ***Except for Thurs, Oct, 28 between 3AM – 8AM when Piazza monitoring by course personnel will be limited*****, if you do not get a response to a Piazza error/clarity question within 30 min. or by when you absolutely need it, read the question again *carefully* then state your assumptions and answer as best as possible.**

Academic Integrity Declaration (AID): Your AID is to be submitted to ***MarkUs*** where we can auto-check correct completion. The integrity of the grade you receive in this course is very important to you and the University of Waterloo. You must read and sign **Mid-AID.txt** (**found on the course Testing/Exams webpage** https://student.cs.uwaterloo.ca/~cs240/f21/exams.phtml) ***before*** *starting work on the assessment* and submit it to **MarkUS as soon as possible** or Oct. 28 10:00AM Waterloo, Ontario time at the latest **or your assessment will not be marked**. The agreement will indicate what you must do to ensure the integrity of your grade.

Your answers to the assessment questions are to be submitted to ***Crowdmark***. *When the assessment availability has started*, you will receive email from the Crowdmark mailer with the link for submitting your files (if you do not see it, check your spam and junk folders in your email reader). **NOTE: our timer overrides Crowdmark's timer.**

File Format: Submitting to Crowdmark may be done as separate question files (one for each question) **or** *one* file, **as long as**, each question answer starts on a new page so you can drag page by page to the correct question in Crowdmark. Crowdmark accepts PDF, JPG, and PNG files. The size limit is 12mb per JPG or PNG file and 25mb per PDF file.

Make sure each page has the question number and part labels (if any). You may use LaTeX to create your file or hand write your answer and scan it or take a picture.

Generally speaking, you shouldn't need more than one page for a question answer. However, you can still submit more than one page per question, if necessary (e.g., your writing is large oryou include a drawing).

Double check your submitted files *before* the deadline to make sure you submitted the intended files, that each answer is submitted to the proper question and that your solution is legible (we can't mark what we can't read). Submit at least 5 minutes early to allow time for this.

Make sure to allocate **enough time to create the necessary file(s) and submit them before *your* deadline** (*not* Crowdmark's deadline)**.**

Remember, late assessments will not be marked.

More complete instructions for submitting and verifying submission can be found at:
https://crowdmark.com/help/completing-and-submitting-an-assignment/
https://crowdmark.com/help/verifying-that-an-assignment-was-submitted/

CS 240 Fall 2021 Midterm Problems

Problem 1 [2+2+2=6 Marks]

a) Suppose you are given a square matrix (same number of rows and columns) of n unordered integers stored as a 2D array. Given a row index i, state the best-case and worst-case runtimes (using the most appropriate order notation) to find the column index j of the first occurrence of a given integer k.

b) Both heaps and AVL trees have an underlying structure that is a binary tree. Briefly explain why the binary tree for a heap is implemented with an array while an AVL tree is typically implemented using a dynamically allocated linked node structure.

c) Suppose we perform MSD-Radix-Sort on the following set of numbers using a radix of 100, where each base 100 digit is represented as two base 10 digits.

$[4586, 3786, 2358, 496, 2335, 38]$

Give the array of integers resulting from the first call to bucket-sort.
State the total number of calls to bucket-sort required to sort the array above and the depth of recursion (count the initial call to start sorting as depth 1).

Problem 2 [2+3=5 Marks]

a) Given a random BST on n distinct items, state the expected runtime to search for a key k. Briefly justify your answer.

b) Explain why any comparison based algorithm to sort an array of size 5 must use at least 7 comparisons in the worst case.

Problem 3 [4 Marks]

Heapsort, as described in class, is not stable. Explain how heapsort can be modified so that it is stable; i.e. your algorithm must still use a heap to sort items. Your algorithm must have runtime $O(n \log n)$. Briefly justify your algorithm is correct and has the required runtime.

Problem 4 [3+3+5+2=13 Marks]

a) Prove from first principles that $15n + 7 \in o(n \log n)$.

b) Compare the growth rates of $f(n) = n^{(3/2)}$ and $g(n) = n(log(n))^2$ using the most appropriate asymptotic notation. Justify your answer using any method presented in class.

c) Analyze the pseudocode below and give a Θ-bound on the running time as a function of n.
You may assume that calculating n^i takes $\Theta(\log i)$ time.

```
mystery(n)
1.   m ← 0
2.   for i ← 1 to n do
3.        if i < n do
4.             for j ← 1 to n do
5.                  for k ← i to j do
6.                       m ← m + 1
7.        else
8.             j ← n^i
9.             while j ≥ 1 do
10.                 j ← j/4
11.                 m ← m + 1
```

Informally but thoroughly justify your bound; i.e. a complete mathematical proof is not required.

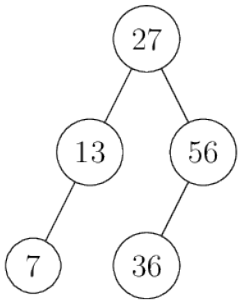
d) Consider the following algorithm.

```
luckyseven(n)
1.   i := random outcome from roll of the die (see below)
2.   if i > 5 do
3.        for j ← ⌈i/2⌉ to (i * n) do
4.             write * to output
5.   else
6.        write ******* to output // writes seven *'s
```

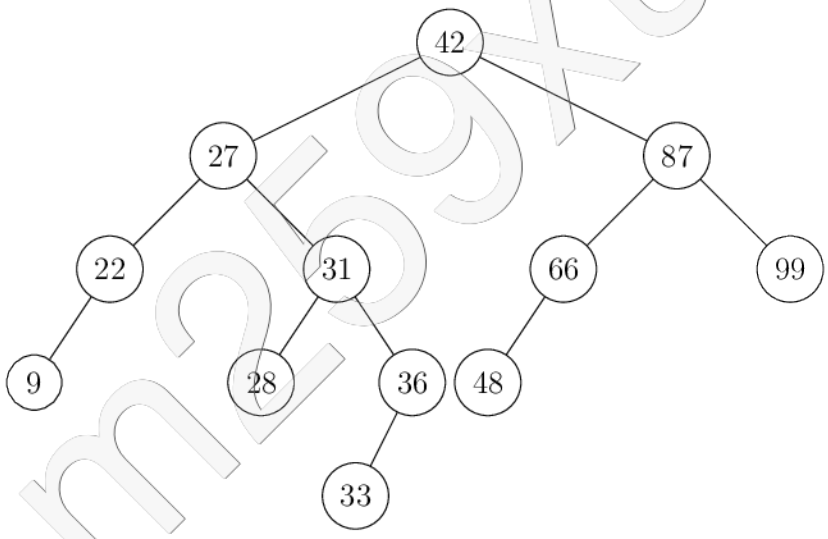
The die used in the function above on line 1 is a 7-sided die (with numbers 1 2 3 4 5 6 7) where each outcome is equally likely. Given an input $n = 10$, state the expected number of times an asterisk (*) will be printed. Show your work.

Problem 5 [2+2=4 Marks]

a) Insert the balance factors in the tree below, then perform an AVL insertion of key 42. Show the resulting tree **and balance factors**.



b) The following is an AVL tree. Perform an AVL deletion on 99. Show the resulting tree **and balance factors**.



Problem 6 [4 Marks]

Suppose you are given a function *find-median* that runs in $\Theta(n^2)$ and finds the median element of a given unsorted array. Formally, analyze the worst-case runtime of QuickSort where *find-median* is used to choose the pivot.

Problem 7 [3+1+2=6 Marks]

a) Draw the skip list that results from inserting $[7, 42, 27, 31, 10]$, in the order given, into an initially empty skip list. Tower height is determined by rolling a 6-sided die and counting the number of 6s rolled in a row. For example, rolling 6 6 3, results in a tower of the same height as flipping H H T (in the algorithm from class), and rolling a 4 results in a tower of the same height as flipping T. Use the following sequence of dice rolls to calculate tower heights:

4 6 6 2 3 6 6 6 5 6 4 6 6 1 5 6 2

Each number may only be used once, in order and some numbers may be unused.

Fill in the table below with the tower heights for each key:

Key	7	42	27	31	10
Height					

b) What is the probability that a key will have a tower of exactly height 3?

c) Suppose that the skiplist contains 1316 keys. Give an estimate on the number of levels expected in the skiplist. Show how you obtained your answer.

Problem 8 [6 Marks]

Suppose we want to create a data structure that stores $(key, value)$ pairs.

Describe a data structure that supports the following operations in O(log(n)) time:

- search(k) returns the corresponding value of key k,
- insert(k, v) inserts pair with key k and value v into the data structure,
- delete(k) deletes the pair with key k
- deleteLast() deletes the most recently inserted pair remaining (i.e. not already deleted) in the data structure.

You may assume that *key*s and *value*s take $O(1)$ space, but this may not hold for other data (particularly unbounded data). You are only allowed to use $O(n)$ space where n is the number of valid elements in the data structure.

You may use data structures and functions defined in class and do not need to restate the implementation or justify their runtimes. If you create your own data structures, briefly justify the operation runtimes.