PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Department of Computer Science

Sample Term Test

CSC 263H1
Duration — 100 minutes

Examination Aids: One 8.5"x11" sheet of paper, handwritten on one side.

The questions in this sample test and their solutions have been collected from tests and exams of past offerings of the course. Therefore, the style and approaches used in the sample solutions might slightly be different than what we did in class.

IMPORTANT NOTE: One of the key factors in a test is **time** pressure. You may be able to answer a question when you have an extended amount of time, but not within the time limit the examiner expects. The followings are the time expectations that designers of the questions in this sample test had in mind:

Q1: 10 min

Q2: 10 min

Q3: 25 min

Q4: 20 min

Q5: 25 min

When answering the questions **time yourself**! If you cannot complete a question within the corresponding time limit it means that you are not prepared yet and need to practice more.

Good Luck!

Question 1. [0 MARK]

Part (a) [0 MARK]

What is the *exact* number of key swaps performed by the non-linear-time BUILD-MAX-HEAP procedure (i.e., the one with $\Theta(n \log n)$ worst-case run time) when it is executed on array $A = \langle 25, 6, 19, 9, 8, 15, 22, 7, 2, 1, 3, 4 \rangle$?

Number of key swaps: _____

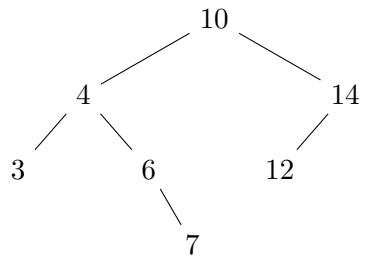
Part (b) [0 MARK]

Perform an INSERT(12) operation on the Max Heap array $A = \langle 25, 9, 19, 6, 8, 15, 4 \rangle$, and *show the resulting array*. Do *not* show any intermediary result.

Question 2. [0 MARK]

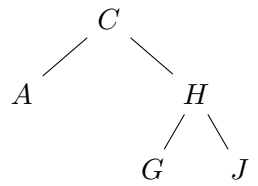
Part (a) [0 MARK]

Insert a node with key 8 into the AVL tree pictured below. Show the intermediate and resulting trees.



Part (b) [0 MARK]

Insert a node with key E into the AVL tree pictured below. Show the resulting tree by either drawing directly on the original tree or by drawing a new tree in the extra space.



Question 3. [0 MARK]

Consider the following Python code which simulates rolling a pair of dice and counting the number of rolls until we reach n pairs. We are interested in the number of time the `random.randint` method is called, which corresponds to the number or rolls. Do not express the complexity in $\mathcal{O}$ notation but give exact expressions.

```
from random import randint

# Attempt to roll n pairs. Use a maximum of 10n rolls.
def countRolls(n):
    count = 0
    tries = 0
    while count < n and tries < 10*n:
        die1 = randint(1,6)      # roll the first die
        die2 = randint(1,6)      # roll the second die
        if die1 == die2:
            count += 1
        tries += 2       # count these 2 rolls even if we don't get a pair
    return count, tries
```

Part (a) [0 MARK]

Perform a worst-case analysis of `countRolls`.

Part (b) [0 MARK]

Perform an average-case analysis of `countRolls`. You do not need to simplify your expressions.

Question 4. [0 MARK]

Consider the following problem: *return the k smallest elements in an array A.* The input array A is in arbitrary order and A is allowed to contain duplicate elements.

Write an algorithm that solves this problem **using a max heap**. Briefly argue that your algorithm is correct (no need for a detailed formal argument, just the main ideas behind why it works). Make your algorithm as efficient as possible no need to try to come up with clever tricks, just be careful to avoid inefficiencies.

Give a tight bound on the worst-case running time of your algorithm expressed as a function of k and n (size of A). Justify your answer.

Question 5. [0 MARK]

You are designing an ADT that contains information about students. Each student has a name, an age and a score. In addition to being able to insert and delete students, you must provide the following new operation in worst case complexity $\mathcal{O}(\log n)$ where n is the number of students. You may assume that the ages are unique.

- $FindBestWithinAge(agelimit)$: returns the student with the highest score from all students whose age does not exceed $agelimit$.

You will accomplish this by augmenting an AVL tree.

1. What will you use as the key for the tree?
2. What additional information will you store at each node?
3. Draw a valid AVL tree for the following records showing any additional information.

Name	Age	Score
Don	8	172
Eshan	10	180
Ken	14	190
Kevin	9	165
Matt	11	185
Nick	6	167
Sam	7	169
Scott	12	187

4. Give the algorithm for $FindBestWithinAge(agelimit)$ explaining briefly why it is $\mathcal{O}(\log n)$.
5. What else do you need to be concerned with when you augment this data-structure? Address this concern here.
6. The assumption that we have only one student in each category is ridiculous. What would have to change in the data-structure or algorithms to accommodate non-unique ages? How would those changes affect the runtime of the operations?