CSC 263H5S 2019 Midterm Test
Duration — 50 minutes
Aids allowed: one double-sided 8.5"x11" sheet

Student Number: ____________________

Last Name: ____________________ **First Name:** ____________________

☐ Registered Lecture Section: L0101/L0103 (Dan Zingaro)
☐ Registered Lecture Section: L0102 (Sushant Sachdeva)

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)

This test consists of 3 questions on 10 pages (including this page). When you receive the signal to start, please make sure that your copy is complete. If you write your answer in the space for rough work, indicate clearly what you want marked.

# 1:	______/14
# 2:	______/ 8
# 3:	______/ 6
TOTAL:	______/28

Question 1. [14 MARKS]

Provide a short answer to each of the following questions.

Part (a) [2 MARKS]

The **level-order** traversal of a tree visits the nodes at depth 0, then the nodes at depth 1, then the nodes at depth 2, etc. (That is, it visits the root, then the root's children, then the children of the root's children, etc.)

Consider a binary max-heap with 7 distinct elements. Is it possible that the level-order traversal of this heap visits the elements in reverse-sorted order?

circle Yes or No, then briefly justify your answer.

Circle one: Yes / No

Justification:

Part (b) [4 MARKS]

You are studying a new algorithm that takes a list as input. You have found a list for each n such that the algorithm takes exactly n^2 steps. What does this tell you about the worst-case runtime, in terms of O and Ω? What does this tell you about the best-case runtime, in terms of O and Ω?

Part (c) [3 MARKS]

Consider the following claim on hash tables implemented with chaining: After calling `Insert(1)`, `Insert(2)`, . . . , `Insert(m)` on a chaining hash table of size `m`, the load factor of the table is guaranteed to be 1.0. Is this claim true or false? First, circle True or False, then briefly justify your answer.

Circle one: True / False

Justification:

Part (d) [2 MARKS]

(From PS2) How many AVL-balanced trees are there with 3 total nodes and 1 leaf?

Part (e) [3 MARKS]

Recall the lecture example of expanding dynamic arrays during a sequence of m `APPEND` operations.

Rather than doubling the size of the array when it is full, suppose we instead increase the size by exactly 3 (e.g. from size 3 to size 6, size 6 to size 9, etc.). Starting with an empty array of size 3, what is the total cost of the m `APPEND` operations? Write your answer using Σ notation; don't worry about solving the sum.

Question 2. [8 MARKS]

You'll get 2 out of 8 marks if you leave this question completely blank.

Here is code that you have seen before for searching a linked list. Let n denote the length of `L`. We are interested in the number of times that the `*` line executes.

```
Search42(L):
  z = L.head
  while z != None and z.key != 42: # *
    z = z.next
  return z
```

Input distribution: The sample space consists of all possible lists of length n of **distinct** integers. Assume that the lists are distributed in such a way that `42` is **equally likely** to be in the list or not in the list. If `42` is in the list, then it is **equally likely** to be at any given position in the list.

Now, perform runtime analysis for `Search42` by answering the following questions. All your answers must be in exact forms rather than in asymptotic notations.

Part (a) [1 MARK]

In the best case, what is the number of comparisons made by `Search42`?

Part (b) [1 MARK]

What is the probability of the best case?

Part (c) [6 MARKS]

In the average case, what is the expected number of comparisons made by `Search42`? Show detailed steps of your calculation, and solve any sums.

Question 3. [6 marks]

You are given a list of distinct integers `lst`, and a target integer `t`. Your goal is to write operation `good_sum(lst, t)` that determines whether or not two different numbers in `lst` sum to exactly `t`. For example:

```
>>> good_sum([4, 2, 6, 5, 10], 7)
True # because 2+5 = 7
>>> good_sum([4, 2, 6, 5, 10], 13)
False # no two numbers in the list add to 13
```

`good_sum` is required to run in average-case $O(n)$ time, where n is the number of integers in `lst`.

Write pseudocode below for `good_sum`. Then, briefly explain why your algorithm is correct and how it achieves the desired runtime. State any reasonable assumptions that you make.

[This page will be marked only if you clearly indicate the part that should be marked.]

[This page will be marked only if you clearly indicate the part that should be marked.]

[This page will be marked only if you clearly indicate the part that should be marked.]

Last Name: ________________________ **First Name:** ________________________