CSCA48 Summer 2017 Final Exam
Duration — 2 hours 50min
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_| **A**

Markus Login: ____________________

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

Question 0. [1 MARK]

Carefully read and follow all instructions on this page, and fill in all fields.

*Do **not** turn this page until you have received the signal to start.*

This exam consists of 5 questions on 14 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Proper documentation is required for all functions and code blocks. If you use any space for rough work, indicate clearly what you want marked. Please read all questions thoroughly before starting on any work. Any pages detached from this cover page will not be marked.

The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University.

# 0:	______/ 1
# 1:	______/ 4
# 2:	______/10
# 3:	______/ 5
# 4:	______/10
# 5:	______/10
TOTAL:	______/40

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 1. [4 marks]

Part (a) [2 marks]

Briefly explain why quicksort is called quicksort if we saw in lecture that it is $O(n^2)$ while other sorting algorithms are $O(nlog(n))$

Part (b) [2 marks]

Briefly explain why representation invariants are important in modern software engineering.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 2. [10 MARKS]

For each of the following scenarios, give the lowest possible big-oh boundary.

a) Finding the smallest value in an unsorted list of size n

b) Finding the largest value in an unsorted linked-list of size n

c) Finding the largest value in a sorted list of size n

d) Finding the smallest value in a binary search tree with n nodes

e) Finding the smallest value in a max-heap with n nodes

f) Turning a list of n numbers into a binary tree

g) Turning a list of n numbers into a binary search tree

h) Turning a list of n numbers into a heap

i) Solving the three-stool tower of hanoi problem with an n-level tower

j) An algorithm that takes an input of size n, and in the worst case scenario takes $35n+21n^2+log_2(n^6)+1.01^n+0.1n^5$ steps to complete

```
def mystery(x, y, r, d):
    draw_circle(x, y, r)
    if(d > 0):
        if(d%2 == 0): #<-- if d is even
            x1 = x + r/2
            x2 = x - r/2
            y1 = y
            y2 = y
            r1 = r/2
            r2 = r/2
        else:
            x1 = x
            x2 = x
            y1 = y + r/2
            y2 = y - r/2
            r1 = r/2
            r2 = r/2
        mystery(x1, y1, r1, d-1)
        mystery(x2, y2, r2, d-1)

if(__name__ == "main"):
    mystery(0, 0, 100, 3)
```

Question 3. [5 marks]

In the space below, draw the output of the code on the previous page, assuming that `draw_circle(x, y, r)` draws a circle of radius `r` centred at `(x, y)`. Indicate on the drawing which circle is drawn first, and which is drawn last. You don't have to be precise in your drawing, as long as the pattern is correct.

```
                                      |
                                      + 100
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
-100                                  |                                  100
 -+-----------------------------------+-----------------------------------+-
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      + -100
                                      |
def is_min_heap(root):
container = Container()
container.put(root)
container.put(test_node.left)
container.put(test_node.right)
found_error = False
found_error = True
if(is_min_heap(test_node.left) == False):
if(is_min_heap(test_node.right) == False):
if(test_node.left is not None):
if(test_node.right is not None):
if(test_node.left.data < test_node.data):
if(test_node.right.data < test_node.data):
return not found_error
test_node = container.get()
test_node = root
while(not container.is_empty() and not found_error):
```

Question 4. [10 marks]

After re-assembling his code from the previous term test (with your help), Brian decided to make another version of his code to verify if a binary tree is a valid heap, but this time using recursion. Obviously the code mangler doesn't like to be bested, because he/she once again broke into Brian's computer and this time scrambled the code for the two functions together. Mixing up lines, deleting internal comments and duplicate lines. Your job is to re-assemble the two functions from the scrambled code in the space below. Brian can retrieve the docstring from his previous code, so you don't need to reproduce that here.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 5. [10 MARKS]

In lecture, we learned of a way to represent a heap as a list that relied upon the tree structure being binary and complete. Brian thinks he has an algorithm that would let him do the same with a binary search tree. The algorithm is based on inserting and deleting, so he will write those, but he needs you to write the `__init__` and `search` methods in the space below. The `search` method should return the depth in the tree of the searched for value, or -1 if the value is not in the tree). Your code must be recursive (you may not use any loops anywhere in your code), and don't forget documentation (including a representation invariant).

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Bonus. (CONTINUED)

Predict your score on each question of this exam: Get within 10% to get this bonus mark.

Q0:	/1
Q1:	/4
Q2:	/10
Q3:	/5
Q4:	/10
Q5:	/10

Bonus. (CONTINUED)

What would you recommend we do to change/update/improve this course in the future?

Bonus. (continued)

If you could send a message back in time to yourself on the first day of term. What would that message be (no winning lottery numbers allowed)? ‘

Bonus. (continued)

Draw something/write something/tell us a joke. Make at least one TA laugh or smile to get this bonus mark.