UNIVERSITY OF TORONTO
Faculty of Arts and Science
AUGUST 2015 EXAMINATIONS
CSC148H1Y
Duration: 3 hours
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|

Instructor: Brian Harrington

Last Name: _____

First Name: _____

Markus Login: _____

---

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

This exam consists of 7 questions on 18 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.

We have provided you with grids for your answers, this is simply to help you show the indentation of your code and you are not required to adhere to the grids in any specific way.

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.
Please note that you must receive a score of 40% or higher on this exam in order to pass the course.

# 1:	_____/10
# 2:	_____/ 5
# 3:	_____/ 5
# 4:	_____/10
# 5:	_____/10
# 6:	_____/ 5
# 7:	_____/ 5
TOTAL:	_____/50

[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. [10 MARKS]

Write the output of the following code in the space provided.

```
def mystery(s):
    if(len(s) < 2):
        return s[0]
    else:
        m = len(s)//2
        print(">",s[:m], s[m:])
        s1 = mystery(s[:m])
        s2 = mystery(s[m:])

        if(s1[0] == s2[0]):
            print("A: ", s1, s2)
            r = s1 + s2
        elif(s1[0] < s2[0]):
            print("B: ", s1, s2)
            r = s1[0]
        else:
            print("C: ", s1, s2)
            r = s2[0]
        print("<", r)
        return r

result = mystery("TORONTO")
print("result = ", result)
```

[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. [5 MARKS]

Assuming you have `draw_triangle` and `get_midpoint` functions which work as expected, draw the result of calling `crazy_triangle(A, B, C, 4)` in the space below. Indicate which triangle is drawn first, and which is drawn last.

```
def crazy_triangle(order, p1, p2, p3):

        if order >= 0:
            draw_triangle(p1, p2, p3)
            m12 = get_midpoint(p1, p2)
            m23 = get_midpoint(p2, p3)
            m31 = get_midpoint(p3, p1)
            if(order % 3 == 0):
                crazy_triangle(order - 1, m31, m23, p3)
                crazy_triangle(order - 1, m12, p2, m23)
            elif(order % 3 == 1):
                crazy_triangle(order - 1, p1, m12, m31)
                crazy_triangle(order - 1, m12, p2, m23)
            else:
                crazy_triangle(order - 1, p1, m12, m31)
                crazy_triangle(order - 1, m31, m23, p3)
```

. A

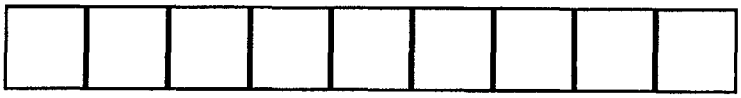
B. .C

[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 3. [5 MARKS]

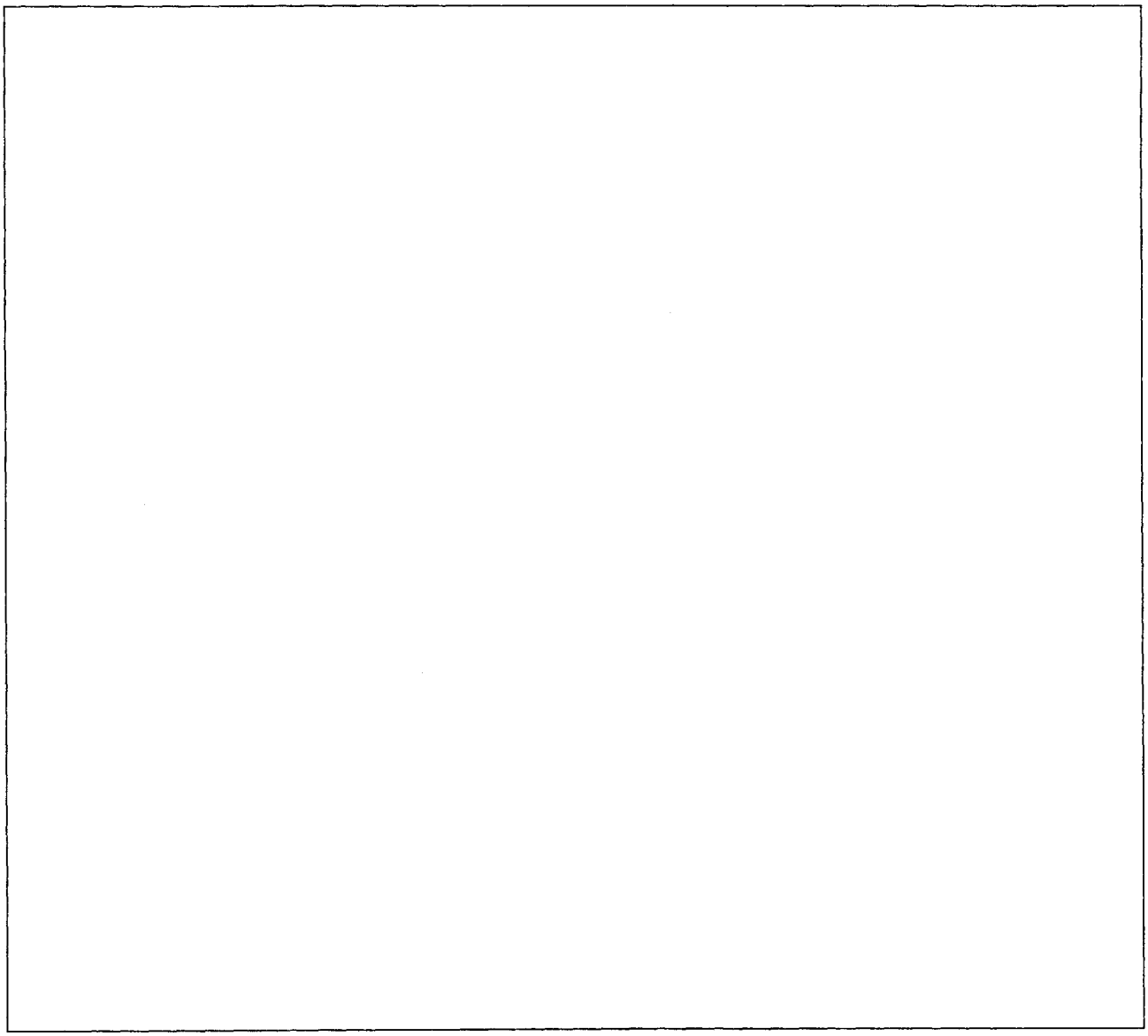
In the space below, draw the list-representation of the max-heap that would be created by the following operations. For full marks, you should show your work:

```
insert(4)
insert(0)
insert(5)
remove_max()
insert(1)
insert(2)
insert(9)
insert(8)
remove_max()
insert(6)
insert(3)
insert(7)
remove_max()
insert(11)
insert(10)
```



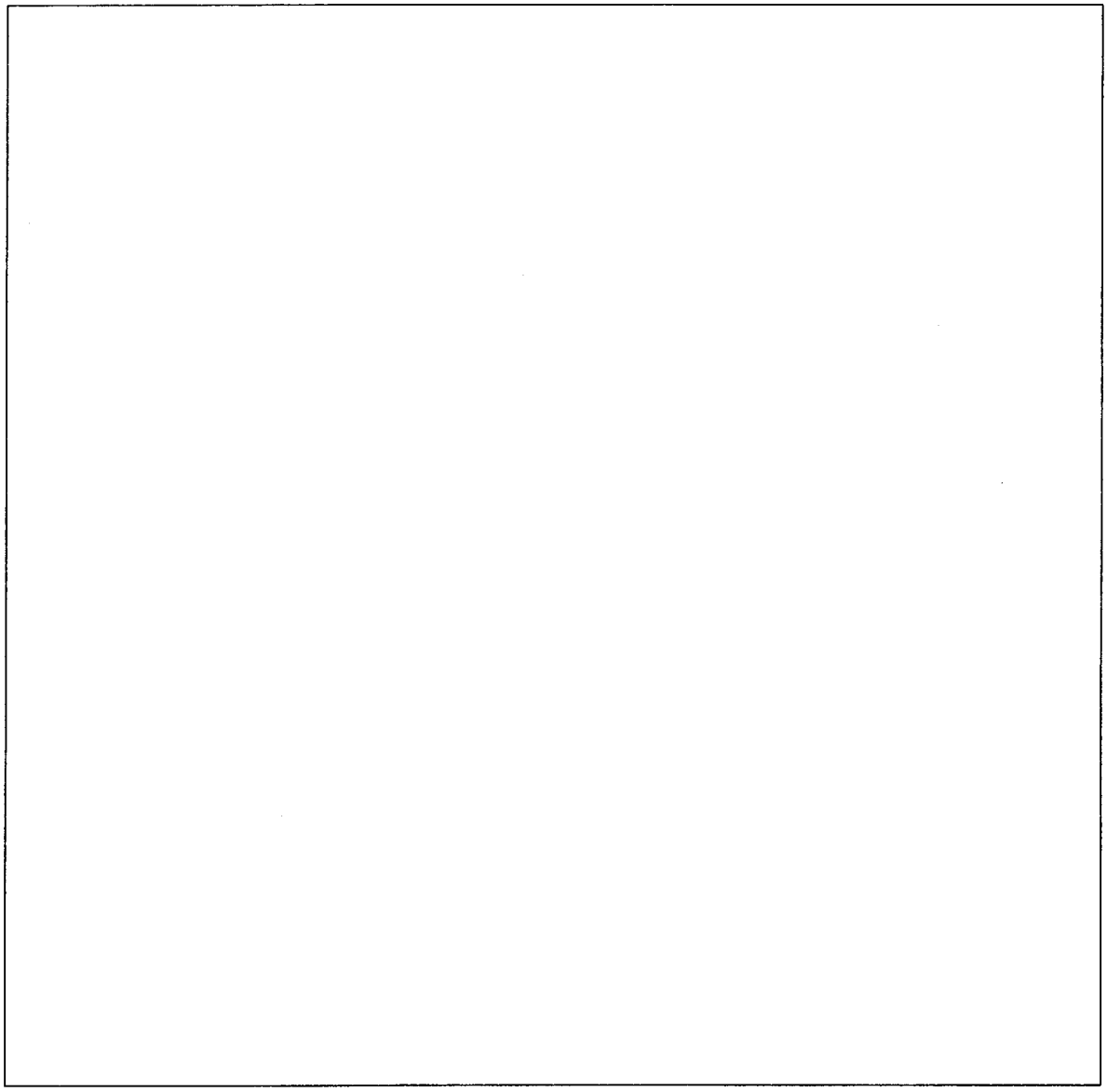
```
class BTNode(object):
    """A node in a binary tree."""

    def __init__(self, value, left=None, right=None):
        """(BTNode, int, BTNode, BTNode) -> NoneType
        Initialize this node to store value and have children left and right,
        """
        self.value = value
        self.left = left
        self.right = right
```



Question 4. [10 MARKS]

Define the `balance` of a tree to be the total number of nodes in the tree which are right children minus the total number of nodes in the tree which are left children. In the space below, write the `BTNode` method `balance` which returns the balance of the tree rooted at that node. The `BTNode` class can be found on the previous page.



[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 the space below, complete the following method. You may use helper functions if you wish, but for full marks your code must be efficient (use minimal extra space, and only traverse the tree once), and leave the tree un-changed). The `BTNode` class is is the same one used in the previous question.

```
def second_deepest(self):
    '''(BTNode) -> int
    Return the depth of the second deepest node in the tree rooted at this
    node. The depth of the root node is assumed to be 0
    RAISES: SmallTreeError if there are fewer than 2 nodes in this tree
    '''
```

[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 6. [5 MARKS]

Put an x to indicate the asymptotic upper bound (Big-Oh) for the following operations. Assume that each data structure contains n values:

	$O(1)$	$O(n)$	$O(log(n))$	$O(n * log(n))$	$O(n^2)$
Searching for a number in an unsorted list of numbers					
Searching for a number in a sorted list of numbers					
Searching for a number in a Binary Search Tree					
Searching for a number in a Heap					
Inserting a number into a Binary Search Tree					
Inserting a number into a Linked List of numbers					
Inserting a number into a Heap					
Inserting a number into an n x n Matrix (from A1)					
Sorting a list of numbers using InsertionSort					
Sorting a list of numbers using a Heap					

Question 7. [5 MARKS]

Are the following statements True or False? Justify your answer:
$25n^2 + 15n + 7 \in O(n^2)$

$12n * log(n) + 21n + 150 \in O(n^2)$

[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.

Write the names of all TAs for this course (1 mark for each TA correctly named)

Bonus.

Write a python function that does something interesting. Anything at all. Have fun with it.

Bonus.

Bonus mark for reading all the questions before you start writing... don't do the first bonus question (-1 mark for each TA name written on the previous page)

Bonus.

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.

Tell us something you learned this term that had nothing to do with this course

Bonus.

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

Short Python function/method descriptions:

You may tear this page off, but if you do so, you must not include any work on it (front or back) that you wish to have marked.

```
__builtins__:
  abs(number) -> number
    Return the absolute value of the given number.
  max(a, b, c, ...) -> value
    With two or more arguments, return the largest argument.
  min(a, b, c, ...) -> value
    With two or more arguments, return the smallest argument.
  isinstance(object, class-or-type-or-tuple) -> bool
    Return whether an object is an instance of a class or of a subclass thereof.
    With a type as second argument, return whether that is the object's type.
  int(x) -> int
    Convert a string or number to an integer, if possible.  A floating point argument
    will be truncated towards zero.
  str(x) -> str
    Convert an object into a string representation.

str:
  S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
  S.find(sub[,i]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
  S.isalpha() --> bool
    Return True if and only if all characters in S are alphabetic
    and there is at least one character in S.
  S.isdigit() --> bool
    Return True if and only if all characters in S are digits
    and there is at least one character in S.
  S.islower() --> bool
    Return True if and only if all cased characters in S are lowercase
    and there is at least one cased character in S.
  S.isupper() --> bool
    Return True if and only if all cased characters in S are uppercase
    and there is at least one cased character in S.
  S.lower() --> str
    Return a copy of S converted to lowercase.
  S.replace(old, new) -> str
    Return a copy of string S with all occurrences of the string old replaced
    with the string new.
  S.split([sep]) -> list of str
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
  S.startswith(prefix) -> bool
    Return True if S starts with the specified prefix and False otherwise.
  S.strip() --> str
    Return a copy of S with leading and trailing whitespace removed.
  S.upper() --> str
    Return a copy of S converted to uppercase.
```

```
list:
  append(...)
    L.append(object) -- append object to end
  count(...)
    L.count(value) -> integer -- return number of occurrences of value
  index(...)
    L.index(value, [start, [stop]]) -> integer -- return first index of value.
    Raises ValueError if the value is not present.
  insert(...)
    L.insert(index, object) -- insert object before index
  pop(...)
    L.pop([index]) -> item -- remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.
  remove(...)
    L.remove(value) -- remove first occurrence of value.
    Raises ValueError if the value is not present.

math:
   ceil(...)
        Return the ceiling of x as an int.
        This is the smallest integral value >= x.
    cos(...)
        Return the cosine of x (measured in radians).
    floor(...)
         Return the floor of x as an int.
        This is the largest integral value <= x.
    pow(...)
        Return x**y (x to the power of y).
    sin(...)
        Return the sine of x (measured in radians).
    sqrt(...)
        Return the square root of x.
    tan(...)
        Return the tangent of x (measured in radians).
set:
  pop(...)
    Remove and return an arbitrary set element.
    Raises KeyError if the set is empty.
dict:
  keys(...)
    D.keys() -> a set-like object containing all of D's keys
  get(...)
     D.get(k[,d]) -> returns D[k] if k is in D, otherwise returns d.  d defaults to None.
object:
  __init__(...)
    x.__init__(...) initializes x; called automatically when a new object is created
  __str__(...)
    x.__str__() <==> str(x)
other:
   x // y = integer divide x by y (i.e., how many times does x divide evenly into y). 5 // 3 = 1
   x % y = the remainder when x is integer divided by y. 5 % 3 = 2
```