UNIVERSITY OF TORONTO
Faculty of Arts and Science

CSC148H1Y

AUGUST 2012 EXAMINATION
Duration – 3 hours
No Exam Aids Allowed

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

Family Name: ____________________ Given Name: ____________________

Do **not** *turn this page until you have received the signal to start.*
(Fill out the identification section above and **read the instructions below.**)
Good Luck!

— This test consists of 7 questions on 12 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
— You don't have to write any docstrings, unless a question explicitly asks for them.
— If you are unable to answer a question (or part), you will get 20% of the marks for that question (or part) if you write "I don't know" and nothing else. You will *not* get those marks if your answer is completely blank, or if it contains contradictory statements (such as "I don't know" followed or preceded by parts of a solution that have not been crossed off.
— Unless otherwise specified, helper methods and functions are always allowed.
— If you use any space for rough work, indicate clearly what you want marked.

# 1:	______/ 6
# 2:	______/14
# 3:	______/ 8
# 4:	______/12
# 5:	______/ 8
# 6:	______/14
# 7:	______/14
TOTAL:	______/76

Question 1. [6 MARKS]

This question is about **object oriented programming.**

You are given a Queue class (no need to import it), with enqueue, dequeue, and is_empty methods. Your task is to implement a PriorityQueue class that is is optimized for a particular situation: only 10 possible priority values (the integers from 0-9). You should implement this by maintaining 10 different queues, one for each priority value. The __init__ method has been defined for you. Complete the insert and get_min methods below. If the priority given to insert is not between 0 and 9, you should raise a ValueError.

```
class PriorityQueue(object):
    def __init__(self):
        self._qs = []
        for i in range(10):
            self._qs[i] = Queue()

    def insert(self, e, priority):
        """(PriorityQueue, object, int) -> NoneType
        Insert e into the priority queue, given an associated priority.
        Raises a ValueError if priority is not in [0, 9]
        """







    def get_min(self):
        """PriorityQueue -> object
        Remove and return the object with the lowest priority.
        """
```

Question 2. [14 MARKS]

This question is about **linked lists**.

The parts on this page refer to a linked list class that maintains a reference to just the first (head) node in the list. Answer the following True/False questions based upon this linked list class.

Part (a) [2 MARKS]

Removing a node from the middle of the linked list is faster (in practice) than adding a node to the end.

☐ True

☐ False

Part (b) [2 MARKS]

This linked list class can be used to implement an efficient queue (enqueue and dequeue operations are fast regardless of the list size).

☐ True

☐ False

Part (c) [2 MARKS]

Storing a reference to the tail (last node) of the linked list would make it much more efficient to remove nodes from the tail of the list.

☐ True

☐ False

(Question continued on next page.)

Part (d) [8 MARKS]

Draw the linked list structure that the variable a points to at three locations (A, B, and C) in the following program. You should follow the form we used in lecture, with each Node represented as a rectangle with two cells, one for each instance variable (the left for data and the right for next).

```
class Node(object):
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

def foo(node):
    if node is None:
        return 0
    else:
        node.data += foo(node.next)
        return node.data

a = Node(4, Node(3, Node(2, Node(1))))
########## A ##########
foo(a)
########## B ##########
a.next.next = a.next.next.next
########## C ##########
```

Draw the structure that a points to at line A:

Draw the structure that a points to at line B:

Draw the structure that a points to at line C:

Question 3. [8 MARKS]

This question is about **inheritance**.

What is the output of the following code:

```
class A(object):
    def __init__(self):
        self.a = 0
        self.b = 1

    def m1(self):
        print "A m1 %d" % self.a

    def m2(self):
        self.a += 1
        print "A m2 %d" % self.b
        self.m1()

class B(A):
    def __init__(self):
        self.a = 42
        A.__init__(self)

    def m1(self):
        print "B m1 %d" % self.a
        self.b += 2

b = B()
# What is the output of the following four lines?
b.m1()
b.m2()
print "a %d" % b.a
print "b %d" % b.b
```

Write your answer here:

Question 4. [12 MARKS]

This question is about **sorting**.

You've passed CSC148, completed your degree, and are now consulting for a local tech start-up that deals with lots of different datasets. They want your advice on which sorting algorithm to use for each type of dataset.

The sorting algorithms we've covered in class are: bubble sort, insertion sort, selection sort, heapsort, mergesort, quicksort, and radix sort. For each of the following, **choose the best sorting algorithm** for the dataset (from the ones we've covered) and **briefly explain** why that sorting algorithm is the best (one sentence at most). If there are multiple "best" algorithms, pick only one.

Part (a) [3 MARKS]

One dataset is a very large database of voltage measurements (float values in a wide range) from weather sensors from across Canada. They don't know much else about the input data. What sorting algorithm should they use to sort all the voltages?

Part (b) [3 MARKS]

One dataset consists of thousands of small files, each with 10 lines. Each file corresponds to the elevation (in meters) of a climber ascending a mountain, taken at 10 time points. The first data point is a measurement of the elevation at the base camp and the last data point is measurment of the elevation at the summit. The measurements are noisy and some parts of the climb are downhill. The company would like to sort the values in each file separately (to make it easy to later calculate min, max, and median elevation, as well as elevation gain). Which sorting algorithm should they use to sort the values in each file?

Part (c) [3 MARKS]

One dataset contains hundreds of thousands of students and their associated standardized test scores. They want to sort by the test scores, but they just want to get the top k students efficiently, for some reasonably small number k (and they don't need the rest of the dataset to be sorted). Which sorting algorithm should they use?

Part (d) [3 MARKS]

One census dataset consists of the age (in years) of every individual in Canada and the USA (about 350 million integers). Which sorting algorithm should they use?

Question 5. [8 MARKS]

This question is about **sorting**.

Here is a sorting algorithm that we will call **meansort** (it came up in a question in lecture). Pseudo-code for the algorithm is as follows:

```
meansort(L):
    if size(L) > 1:
        calculate m, the mean of the values in L
        create a list, L2, of the values in L < m
        create a list, L3, of the values in L >= m
        L[:len(L2)] = meansort(L2)
        L[len(L2):] = meansort(L3)
```

Part (a) [2 MARKS]

This algorithm is very similar to another sorting algorithm we studied in class: **quicksort**. What is the primary difference?

Part (b) [4 MARKS]

In what ways is this sorting method better than **quicksort**? In what ways is it worse? (*e.g.* performance, specific inputs, etc.)

Part (c) [2 MARKS]

With respect to n, how many operations will this algorithm take to sort a list of size n? (What is the worst-case runtime of this algorithm, where n is the number of elements in the list?)

- ☐ $\mathcal{O}(\log n)$
- ☐ $\mathcal{O}(n)$
- ☐ $\mathcal{O}(n \log n)$
- ☐ $\mathcal{O}(n^2)$

Question 6. [14 MARKS]

This question is about **binary heaps** (and complete binary trees).

Parts (a) through (c) refer to this list: `[1, 3, 7, 5, 4, 9, 6, 8]`

Part (a) [2 MARKS] Interpreting the list as a complete binary tree, what is the `preorder` traversal of the tree?

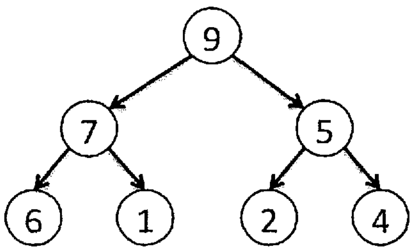
Part (b) [2 MARKS] What is the height (in nodes) of this binary tree?

Part (c) [2 MARKS] The list corresponds to a valid binary min-heap.

☐ True

☐ False

Parts (d) and (e) refer to the following valid binary max-heap:



Part (d) [4 MARKS] Perform a `remove_max` operation on the max-heap and draw the resulting heap:

Part (e) [4 MARKS] Insert 9 into the max-heap and draw the resulting heap:

Question 7. [14 MARKS]

This question is about **binary trees**.

This question uses the following `Node` class:

```
class Node(object):
    def __init__(self, data, left=None, right=None):
        self.data = data
        self.left = left
        self.right = right
```

Part (a) [8 MARKS]

Define a **recursive** function `trees_same_shape` that takes two `Node` objects (the roots of two different binary trees) as input and returns True iff the trees have the same "shape". Two trees have the same "shape" if they have nodes in exactly the same places, ignoring the values in the nodes. For example, it should return True given the roots of (A) and (B), but False given the roots of (A) and (C):

```
 (A)         (B)         (C)

  1           6           1
 / \         / \         / \
2   3       7   8       2   3
   / \         / \         /
  4   5       9  10       4
```

You are **not** permitted to use any helper functions to solve this. Either argument may be `None`, corresponding to an empty tree.

(Question continued on next page.)

The following is an iterative function that can, with very minor changes, generate either a preorder, inorder, or postorder traversal of a binary tree of Nodes:

```
def traverse(root):
    s = Stack()
    s.push((root, False, False))

    while not s.is_empty():
        node, first, second = s.pop()
        if not first and not second:
            print node.data  ######## A ########
            s.push((node, True, False))
            if node.left is not None:
                s.push((node.left, False, False))

        elif first and not second:
            print node.data  ######## B ########
            s.push((node, True, True))
            if node.right is not None:
                s.push((node.right, False, False))

        else:
            print node.data  ######## C ########
```

You get to choose to leave everything as is, or to change any or all of the `print` lines (A, B, and/or C) to `pass`.

Part (b) [6 MARKS]

Which lines, if any, of A, B, and/or C, would you change to `pass` to make `traverse` print a **preorder** traversal?

Which lines, if any, of A, B, and/or C, would you change to `pass` to make `traverse` print an **inorder** traversal?

Which lines, if any, of A, B, and/or C, would you change to `pass` to make `traverse` print a **postorder** traversal?

Use this page for rough work and for answers that didn't fit. Indicate clearly what you want us to mark.

Use this page for rough work and for answers that didn't fit. Indicate clearly what you want us to mark.