PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2009 EXAMINATIONS

CSC 148 H1Y
Instructor(s): R. Danek

Duration — 3 hours

Examination Aids: None.

Student Number:	
Last (Family) Name(s):	
First (Given) Name(s):	

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

Instructions:

- Check to make sure that you have all 20 pages.
- Read the entire exam before you start.
- Not all questions are of equal value, so budget your time accordingly.
- You do not need to add `import` lines or do error checking unless explicitly required to do so.
- You do not need to write docstrings or comments unless explicitly required to do so, although it may help get you part marks if your answer is otherwise incorrect. Also, we have to be able to understand your code. If you feel you are doing something subtle or confusing in your code, you should briefly explain it with a comment.
- If you use any space for rough work, indicate clearly what you want marked.

MARKING GUIDE

# 1:	______/ 8
# 2:	______/ 10
# 3:	______/ 14
# 4:	______/ 20
# 5:	______/ 14
# 6:	______/ 12
# 7:	______/ 10
# 8:	______/ 12
TOTAL:	______/100

Good Luck!

Question 1. [8 MARKS]

A friend of yours has information about the order in which nodes are visited in the preorder, inorder, and postorder traversals of a binary search tree. He wants to reconstruct the drawing of the original tree from this information, but isn't sure how. He knows that you have taken CSC148 and are able to help him, so he emails you a copy of the traversals.

Unfortunately, as the email is being transmitted to your computer, a network glitch occurs and some of the characters in the email become garbled. You're about to ask your friend to resend the traversals, when you realize that despite some of the characters being garbled, there's still sufficient information to reconstruct the original tree in its entirety.

Draw the original binary search tree below using the information provided in the traversals. A single garbled character is represented by a single '?' (question mark). Question marks do not represent actual nodes in the tree, and should not appear anywhere in the tree that you draw. You may assume that the tree does **NOT** contain duplicate nodes, and that characters that are not garbled were correctly transmitted. Also, assume that $A < B < C < ... < X < Y < Z$, as in the natural ordering of the alphabet.

```
Preorder : ? A ? X ? ? Z

Inorder   : ? C M ? ? ? ?

Postorder: ? ? T P Z ? ?
```

Question 2. [10 MARKS]

You are given the following `BinaryTree` class:

```
class BinaryTree:
    def __init__(self, key):
       self.key = key
       self.left = None
       self.right = None
```

Write a function `convert(L)` that takes as an argument `L`, a flat (i.e., unnested) python list that represents a **complete** binary tree. If `L` is empty, then `convert` should return `None`. Otherwise, `convert` should return an instance of the `BinaryTree` class that represents a tree equivalent to the tree represented by `L`.

Assume that a python list representation of a complete binary tree starts at index 0, as we studied in class.

You may create a helper function if you wish, but you may not modify the `BinaryTree` class.

```
def convert(L):
```

Question 3. [14 MARKS]

A *Binary Remainder Tree* is a binary tree that satisfies the following properties: (i) Every node's key is an integer; (ii) For each node x, and each node y, if y is in x's left subtree then the remainder of y's key divided by x's key is 0; and if y is in x's right subtree then the remainder of y's key divided by x's key is not 0. (Recall, in python a % b is the remainder of a divided by b.)

Part (a) [6 MARKS]

We provide the following partial implementation of the BinaryRemainderTree class and the RTreeNode class. Complete the implementation of the put method in RTreeNode.

```
class BinaryRemainderTree:
   def __init__(self):
      self.root = None

   def put(self, key, value):
      ''' Insert a new node into the binary remainder tree
      with the given key and value.'''

      if self.root: self.root.put(key,value)
      else: self.root = RTreeNode(key,value)

class RTreeNode:
   ''' Represents a node in BinaryRemainderTree.'''
   def __init__(self, key, value):
      self.key = key
      self.value = value
      self.left = None
      self.right = None

   def put(self, key, value):
      ''' Insert a new node, with the given key and value, into the
      binary remainder tree rooted at the current node.'''
      # TODO: implement
```

Part (b) [2 MARKS]

Using big-oh notation, state the worst-case time complexity for your put method in terms of n, where n is the number of items in the binary remainder tree. Give the strongest answer that you can, that is, the tightest bound. No justification is necessary.

Part (c) [2 MARKS]

Briefly describe in plain English (i.e., don't write python code) how to delete a node from a binary remainder tree assuming that the node has at most one child. Use no more than two or three sentences.

Part (d) [4 MARKS]

Can you delete a node with two children in a binary remainder tree using the same technique that is used for deleting a node with two children in a binary search tree? If yes, then briefly justify why. If not, then draw an example binary remainder tree and use it to illustrate what can go wrong. Assume in your answer that you already have a reference to the node being deleted, i.e., you don't have to search for it. (Thus your answer should not be based on the difference between searching for a node in a binary remainder tree and searching for a node in a binary search tree.)

Question 4. [20 MARKS]

Below are modified versions of the `mergesort` and `merge` functions that we studied in class. The `mergesort` function takes as an argument a python list to be sorted, and the `merge` function takes as arguments two sorted python lists, `left` and `right`, to be merged together into a sorted list. Both functions also have an argument named `choices`. This argument is used to record the sequence of choices made by the `merge` function during sorting. A choice is simply the string "LEFT" or "RIGHT". As the `merge` function executes, if the next item to be merged into the sorted list is from the `left` argument, the choice "LEFT" will be added to the end of the `choices` argument; otherwise, the choice "RIGHT" will be added to the end of the `choices` argument.

```
def mergesort(alist, choices):
   ''' Return a sorted version of alist. The function keeps track of choices
   made by merge in the choices argument. alist must be non-empty. '''

   n = len(alist)
   if n == 1: return alist          # base case
   mid = n/2
   left = mergesort(alist[:mid], choices)
   right = mergesort(alist[mid:], choices)
   return merge(left, right, choices)

def merge(left, right, choices):
   ''' Return the left and right sorted lists merged together into a newly
   sorted list. The function keeps track of choices made in the choices argument.'''

   result = []
   i = 0
   j = 0
   while i < len(left) and j < len(right):
      if left[i] <= right[j]:
          result.append(left[i])
          i = i + 1
          choices.append("LEFT")
      else:
          result.append(right[j])
          j = j + 1
          choices.append("RIGHT")

   while i < len(left):
      result.append(left[i])
      i = i + 1
      choices.append("LEFT")

   while j < len(right):
      result.append(right[j])
      j = j + 1
      choices.append("RIGHT")

   return result
```

Question is continued on the next page ...

Part (a) [3 MARKS]

The given `mergesort` function does not work when the `alist` argument is empty. We want to fix this problem by replacing the base case with one of the following:

```
if n == 0: return []            # base case
```

or

```
if n <= 1: return alist         # base case
```

Circle the base case above that we should use and briefly explain why that one is the right choice.

Part (b) [4 MARKS]

Consider the following call to `mergesort`:

```
choices = []
mergesort([3,8,9,7,5,4],choices)
```

Write the contents of the `choices` list after `mergesort` returns. (You do not need to surround each element with quotes.)

Part (c) [3 MARKS]

Given some python list, `mylist`, with length n, consider the following call to `mergesort`:

```
choices = []
mergesort(mylist, choices)
```

Using big-oh notation, state in terms of n the worst-case length of the `choices` list after `mergesort` returns. Give the tightest bound that you can. You do not need to justify your answer.

Part (d) [10 MARKS]

Suppose you want to use the information stored in the `choices` list after executing `mergesort` to reconstruct the original unsorted list. Write a function `unsort(alist, choices)` that does this. The function takes as arguments a sorted list `alist` and the list of `choices` that were made when the list was sorted with `mergesort`. The return value from the function should be the original unsorted list as it existed prior to `mergesort` sorting it.

Hint: Write a function `unmerge(alist, choices)` that "unmerges" a sorted list `alist` into two sublists based on information in the `choices` argument, and then use `unmerge` within the `unsort` function.

Your function may modify its arguments during the course of execution.

```
def unsort(alist, choices):
```

(Extra space for question 4, if required.)

Question 5. [14 MARKS]

Part (a) [6 MARKS]

For part (a) only, correct answers are worth 2 marks, incorrect answers receive -1 mark penalty, and no answer is worth 0 marks. (The minimum grade for part (a) is 0.)

Circle True or False for each statement. No justification is necessary.

Let $f(n) = \log_2 1 + \log_2 2 + \log_2 3 + ... + \log_2 (n-1) + \log_2 n$.

(2 marks) $n!$ is $O(2^{f(n)})$.	True	False
(2 marks) $n(n+6)(n+\log_2 n)$ is $O(n \log_2 n)$.	True	False
(2 marks) 5 is $O(1)$.	True	False

Part (b) [2 MARKS]

Using big-oh notation, state the worst-case time complexity of the following function in terms of n, the length of the input list. Give the strongest answer that you can, that is, the tightest bound. You do not have to justify your answer.

Answer:

```
def func(inlist):
    n = len(inlist)
    sz = 1
    while sz < n:
        cur = n
        while cur > 0:
            cur = cur - sz
        sz = sz * 2
```

Part (c) [2 MARKS]

Using big-oh notation, state the worst-case time complexity of the following function in terms of n, the length of the input list. Give the strongest answer that you can, that is, the tightest bound. You do not have to justify your answer.

Answer:

```
def func(inlist):
    n = len(inlist)
    sz = 1
    cur = n
    while cur > 0:
        index = 0
        while index < sz and cur > 0:
            cur = cur - 1
            index = index + 1
        sz = sz * 2
```

Part (d) [4 MARKS]

Using big-oh notation, state the worst-case time complexity of the following function in terms of n, the length of the input list. Give the strongest answer that you can, that is, the tightest bound. You may assume that the `stack` variable is declared in the module scope, and that it is an instance of a `Stack`. Further assume that all methods called on the `stack` instance used in the function take $O(1)$ time. **Briefly justify your answer in plain english, using no more than a few sentences.**

```
def func(inlist):
   '''
   Perform some computation using inlist,
   where inlist is a list of non-negative integers.
   '''

   n = len(inlist)
   stack.push(1)
   sz = 1

   while sz < n:
       sz = stack.peek() + 1
       idx = 0
       while idx < sz:
           stack.push(sz)
           idx = idx + 1

   while not stack.is_empty():
       stack.pop()
```

Question 6. [12 MARKS]

Here is a version of bubble sort that sorts a python list into non-descending order:

```
def bubblesort(alist):
    ''' Sort the python list, alist, in non-descending order. '''

    n = len(alist)
    for passnum in range(n-1):
        for i in range(0, n - passnum - 1):
            if alist[i] > alist[i+1]:
                alist[i], alist[i+1] = alist[i+1], alist[i]
```

You are given classes LinkedList and Node, which are used for implementing a linked list:

```
class LinkedList:
    def __init__(self):
        self.head = None

    def length(self):
        ''' Return the length of the list. '''
        # implementation omitted ..

    # no other methods defined

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

Write a modified version of bubble sort that takes a LinkedList instance as an argument and sorts it into non-descending order. Your version should sort the list in-place, and only require a constant amount of memory in addition to the original memory allocated to the list being sorted. Also, the time complexity of your modified bubble sort should be the same as the original bubble sort.

Note: You **cannot** modify the _data attribute of Node instances. Also, it is strongly recommended that you draw a linked list diagram to help you out.

Complete the implementation on the next page.

```
def bubblesort_ll(alist):
    ''' Sort a LinkedList instance, alist, in non-descending order. '''
```

Question 7. [10 MARKS]

You are given Queue and Stack classes below.

```
class Stack:
    def __init__(self):
        ''' Initialize a new stack instance. '''
        # implementation omitted...

    def push(self, o):
        ''' Push o onto the stack. '''
        # implementation omitted...

    def pop(self):
        ''' Remove and return the top element of the stack.'''
        # implementation omitted...

    def peek(self):
        ''' Return the top element of the stack.'''
        # implementation omitted...

    def is_empty(self):
        ''' Return True if the stack is empty, False otherwise.'''
        # implementation omitted...


class Queue:
    def __init__(self):
        ''' Initialize a new queue instance. '''
        # implementation omitted...

    def enqueue(self, o):
        ''' Add o to the end of the queue. '''
        # implementation omitted...

    def dequeue(self):
        ''' Remove and return the front element of the queue.'''
        # implementation omitted...

    def front(self):
        ''' Return the front element of the queue.'''
        # implementation omitted...

    def is_empty(self):
        ''' Return True if the queue is empty, False otherwise.'''
        # implementation omitted...
```

These are the only methods that you can call on instances of Stack and Queue. There are no other methods defined for these classes. In particular, note that neither class defines the size method.

Question 7 continued on next page ...

Question 7 continued from previous page ...

Suppose all the elements in a list `L1` are enqueued, in order, into an initially empty instance of a `Queue`, `q`, and all the elements in list `L2` are pushed, in order, onto an initially empty instance of a `Stack`, `s`. Write a function `is_equal(q,s)` that returns `True` if the lists `L1` and `L2`, which were used to populate `q` and `s`, are equal, and `False` otherwise.

Your solution must obey a few rules:

- It cannot use any python lists, tuples, or dictionaries.
- It cannot access any attributes associated with `Queue` or `Stack` instances.
- It cannot create any new instances of `Queue` or `Stack`.
- It cannot define any new classes. (For example, you cannot define a `LinkedList` class for use in your solution.)

Your function may modify its arguments during the course of execution. Hint: Use recursion. You may find it useful to write a helper function as well.

```
def is_equal(q, s):
```

Question 8. [12 MARKS]

Assume in the questions below that a python list representation of a complete binary tree starts at index 0, as we studied in class.

Part (a) [6 MARKS]

Assume that we insert the following items into a min heap, in order: 15, 20, 5, 7, 3, 8. Draw the tree representation:

Fill in the Python list that stores this heap:

[, , , , ,]

Part (b) [6 MARKS]

For part (b) only, correct answers are worth 2 marks, incorrect answers receive -1 mark penalty, and no answer is worth 0 marks. (The minimum grade for part (b) is 0.)

Let L be a python list that stores a min heap, and let n be its length. Let x be the largest element in L. Answer each question below independently of the others. That is, any modifications to L in one question should be ignored in the other questions.

Circle True or False for each statement. No justification is necessary.

Element x is in `L[n/2:]`.

True False

Immediately after executing `L.reverse()`, L stores a max heap.

True False

Assume that L stores integers. Immediately after executing `L.append(x+1)`, L still stores a min heap.

True False

[There are no exam questions beyond this point. This page will ***not*** *be marked, unless you clearly indicate the part of your work that you want us to mark.]*

Python Lists "Cheat Sheet"

Assume that x is a python list in the following.

Operation	Explanation
`x.append(elt)`	Append `elt` to the end of `x`.
`x.insert(i,elt)`	Insert `elt` at index `i` in `x`.
`x.pop()`	Remove and return the last item in `x`.
`x.pop(i)`	Remove and return the `i`'th item in `x`.
`len(x)`	Return the number of elements in `x`.
`x[i]`	Return the `i`'th item in `x`.
`x[i:j]`	Slice of `x` from `i` to `j`. (Slice includes item `i`, but excludes item `j`.)
`x[:j]`	Slice of `x` from 0 to `j`.
`x[i:]`	Slice of `x` from `i` to the end of `x`.
`x[:]`	Copy of `x`.
`[y]*n`	Return a list containing `n` copies of `y`.

You may assume the following time complexities for python operations:

Operation	Time Complexity
Appending an item to the **end** of a python list.	O(1)
Removing an item from the **end** of a python list.	O(1)
Inserting an item at the **front** of a python list of length n.	O(n)
Removing an item from the **front** of a python list of length n.	O(n)
Creating a slice with k elements from a python list of length n.	O(k)
`len` function.	O(1)

[Use the space below for rough work.]

*[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.]*

[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.]*

[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.]*

Total Marks = 100