UNIVERSITY OF TORONTO
Faculty of Arts and Science
AUGUST 2014 EXAMINATIONS
CSC 148H1 Y
Duration — 3 hours
Aids allowed: none

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

Lecture Section: L0101 Instructor: Jonathan Lung

---

*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.)
Reminder: You need at least 40% on this exam to pass the course. *Good Luck!*

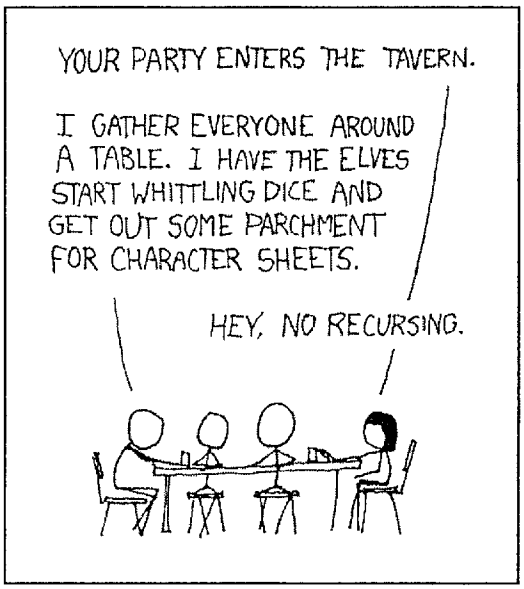
This exam consists of 3 questions on 22 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code. No error checking is required: assume all user input and all argument values are valid.
Any questions where you write "I don't know" or leave blank will receive 10% of the marks for that question. If you write an answer **and** have written "I don't know", we will ignore your answer and treat it as an "I don't know". This applies to the entirety of each part of a question; For example, you cannot answer half of Q1h and write "I don't know" for the second half.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/20

2: _____/25

3: _____/ 5

TOTAL: _____/50



Source: http://xkcd.com/244/

Total Pages = 22

Question 1. [20 MARKS]

Part (a) [1 MARK]

Is learning computer science the same as learning how to program?

Part (b) [1 MARK]

What is a recursive function?

Part (c) [1 MARK]

What is the purpose of a constructor?

Part (d) [1 MARK]

Name a data structure that you could use to implement a queue such that all of the standard queue operations are $O(1)$. If you are thinking of a data structure based on Python's built-in `list`, you need to be more specific than just writing "list".

Part (e) [1 MARK]

When doing object oriented analysis, how do you identify potential classes? How do you identify potential methods? Be sure to specify which part of your answer refers to classes and which to methods.

Part (f) [2 MARKS]

If a piece of code passes an entire set of test cases, can you conclude with absolute certainty that the code behaves as desired? Explain why or why not.

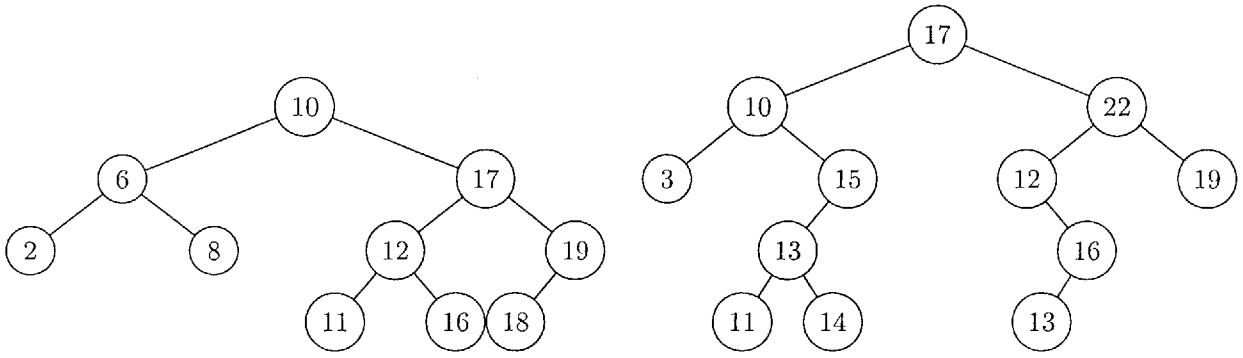
Part (g) [1 MARK]

Without modifying the header below nor using any helper functions, complete the following function according to its docstring. Do not use any loops.

```
1 def sum_to(n: int) -> int:
2     '''Return the sum of integers from 0 through non-negative integer n.
3
4     Examples:
5     >>> sum_to(0)
6     0
7     >>> sum_to(5)
8     15
9     '''
```

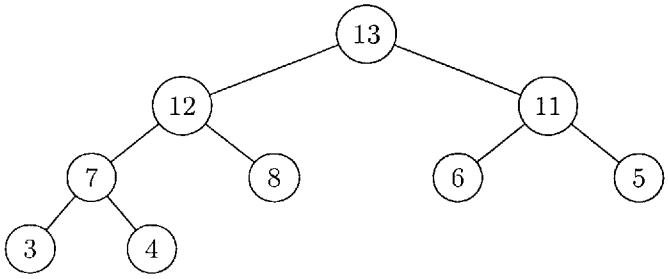
Part (h) [1 MARK]

Draw the result of removing 17 from the binary search tree on the left and the result of removing 17 from the binary search tree on the right. Your answer should thus have two different trees.



Part (i) [2 MARKS]

What kind of abstract data type is the tree below? Be specific. Draw the result of performing a remove operation on it.



Part (j) [1 MARK]

For the original tree in part i above, write out values of the nodes in the order they would be encountered during a preorder traversal starting from the root.

Part (k) [3 MARKS]

Why is the upper bound on the height of a binary heap $\lfloor log_2(n) \rfloor$ where n is the number of nodes in the heap? What is the upper bound for any binary tree and why?

Part (l) [1 MARK]

What is the big-O running time of the following function?

```
1 def filter_a(x: str) -> str:
2     ret_str = ''
3     for ch in x:
4         if ch == 'a':               # True iff ch is the character 'a'
5             ret_str = ret_str + ch
6
7     return ret_str
```

Part (m) [1 MARK]

If a linked list's last node has a reference to the linked list's first node, what kind of linked list is it?

Part (n) [3 MARKS]

Arrange the following three lists in ascending order, indicating equality. For example, if you think $O(x)$ is less than $O(y)$ which is the same as $O(z)$, then write $O(x) < O(y) = O(z)$.

List number one: $O(1), O(2n), O(2^n), O(5), O(n)$

List number two: $O(log_2(n)), O(nlog_2(n)), O(n^2), O(n), O(n^2 + n), O(log_5(n))$

List number three: $O(n^n), O(2^n), O(3^n), O(n^2), O(n^3)$

Question 2. [25 MARKS]

Part (a) [1 MARK]

Compare and contrast stacks and queues by listing at least one similarity and one difference (specifying which is which).

Part (b) [1 MARK]

Compare and contrast `return` and `raise` by listing at least one similarity and one difference.

Part (c) [1 MARK]

Compare and contrast a Python `list` and a linked by listing at least one similarity and one difference.

Part (d) [2 MARKS]

In one short English sentence, describe what the following mystery function does. Then, list a good set of inputs you would use to test the function and the reason for each test.

```
1 def mystery1(x: float) -> int:
2     # Reminder: the min function returns the minimum of its arguments
3     # and max returns the maximum of its arguments.
4     return max(min(x, 100), 0)
```

Part (e) [1 MARK]

In one short English sentence, describe what the following mystery function does.

```
1  def mystery2(n: int) -> bool:
2      '''Precondition: n > 1.'''
3      return _mystery2(n, 2)
4
5
6  def _mystery2(n: int, t: int) -> bool:
7      if t == n:
8          return True
9
10     return (n % t != 0) and _mystery2(n, t + 1)
```

Part (f) [1 MARK]

In one short English sentence, describe what the following mystery function does.

```
1  def mystery3(lst: list) -> None:
2      if len(lst) < 2:
3          return lst
4
5      i = lst.index(max(lst))      # Get the index of the first occurrence of
6                                   # the element with the largest value in lst.
7
8      tmp = lst[i]
9      lst[i] = lst[-1]
10
11     lst.pop(-1)
12     mystery3(lst)
13     lst.append(tmp)
```

Part (g) [1 MARK]

In one short English sentence, describe what the following mystery function does.

```
1  def mystery4(x: int) -> None:
2      if x <= 0:
3          return
4
5      i = 0
6      i = i + 1
7
8      print(i)
9      mystery4(x - 1)
```

Part (h) [1 MARK]

Suppose you are told two algorithms A and B process `lists` in some way and have worst case running times of $O(n)$ and $O(n^2)$, respectively, where n is the length of the input `list`. Can you conclude that if A and B are each given their worst possible inputs for a `list` of any fixed length that A will take less or equal time to run than B? Explain why or why not.

Part (i) [1 MARK]

Consider an empty heap to which we add the ten values 3, 9, 7, 5, 4, 2, 1, 6, 10, 8, in that order. Now, suppose we remove three values from the resultant heap. Which three values are removed from the heap? Specify the order in which they are removed.

Part (j) [1 MARK]

Implement a sorting algorithm of your choice from the following list: bubble sort, insertion sort, selection sort. You may use Python or pseudocode.

Part (k) [1 MARK]

Implement a sorting algorithm of your choice from the following list: merge sort, quicksort, radix sort. You may use Python or pseudocode.

Part (l) [1 MARK]

Complete the `last_node` function according to its docstring. You may not modify the header, but you may add helper functions.

```
1  class LLNode(object):
2      def __init__(self: 'LLNode', value: object, next: 'LLNode') -> None:
3          self.value = value
4          self.next = next
5
6
7  def last_node(start_node: 'LLNode') -> 'LLNode':
8      '''Return the last node in the chain of linked list nodes beginning with
9      start_node.
10     Precondition: start_node is not None.
11
12     Example:
13     >>> last_node(LLNode('a', LLNode('b', LLNode('c', None)))).value
14     'c'
15     '''
```

Part (m) [1 MARK]

Write a function that accepts a DLLNode as an argument and removes it from the linked list to which it belongs. Your function may assume as a precondition that there is at least one node in front of the node passed in as an argument and at least one node after (that is, the DLLNode provided is neither the head nor tail of a linked list).

```
1  class DLLNode(object):
2      def __init__(self, value):
3          self.value = value
4          self.next = None
5          self.prev = None
```

Part (n) [1 MARK]

This question uses the DLLNode defined above in part m. Complete the function below according to its docstring. Do not use any loops.

```
1  def find(node: 'DLLNode', search_value: object) -> 'DLLNode':
2      '''Return the DLLNode object in the same linked list as node whose value
3      matches search_value.
4
5      Examples:
6      >>> head = DLLNode(5)
7      >>> tail = DLLNode(3)
8      >>> head.next = tail
9      >>> tail.prev = head
10     >>> find(head, 3) == tail
11     True
12     >>> find(tail, 5) == head
13     True
14     >>> find(head, 20)
15     None
16     '''
```

Part (o) [1 MARK]

In case you have forgotten how to read a file in Python, the following code prompts a user to enter a file name and prints out the contents of the file:

```
1 filename = input('Enter a file name: ')
2 file_handle = open(filename)
3 file_contents = file_handle.read()
4 print(file_contents)
```

Write a function named `show_file` that prompts the user to enter a file name followed by the contents of the file. The Python `open` function raises a `PermissionError` exception if the user is not allowed to read the file or a `FileNotFoundError` if the file does not exist. In the first case, print out the error message ``Naughty naughty'' and continue to prompt the user to enter a file name (for example, keep prompting forever if the user keeps on entering the names of files s/he is not allowed to read). If the file does not exist, print out the error message ``File not found'' and continue to prompt the user to enter a file name. Do not handle any other exceptions.

Part (p) [1 MARK]

Suppose the following code is in the `__main__` block of your Python program.

```
1 print("You want to see a file?")
2 show_file()
3 print("All done!")
```

What happens if the `read` method raises an `IOError` while calling `show_file`? Include what happens (if anything) in the `show_file` function.

Part (q) [4 MARKS]

Write a class named `Stack` that implements a stack. It should contain a properly named method for determining if it is empty, adding an item, and removing an item. If the method for removing an item is called while the `Stack` is empty, raise an exception named `StackUnderflow` (you will need to create this class). You may write other methods if you wish/need to.

Part (r) [1 MARK]

Complete the count_leaves function below according to its docstring.

```
class BTNode(object):
    def __init__(self, value, left, right):
        self.value = value
        self.left = left
        self.right = right

def count_leaves(root: 'BTNode') -> int:
    '''Return the number of leaves in the tree rooted at root.

    Example:
    >>> count_leaves(None)
    0
    >>> count_leaves(BTNode(
                         0,
                         None,
                         None
                     ))
    1
    >>> count_leaves(BTNode(
                         0,
                         None,
                         BTNode(-3, None, None)
                     ))
    1
    '''
```

Part (s) [1 MARK]

What is the worst-case running time of the following function?

```
1  def reverse(lst: list) -> list:
2      new_list = []
3
4      for item in lst:
5          new_list.insert(0, item)
6
7      return new_list
```

Part (t) [1 MARK]

What is the worst-case running time of the following function?

```
1  def count_numbers(x: str) -> int:
2      num_numbers = 0
3      numbers = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
4
5      for ch in x:
6          for num in numbers:
7              if num == ch:
8                  num_numbers = num_numbers + 1
9
10     return num_numbers
```

Part (u) [1 MARK]

What is the worst-case running time of the following function?

```
1  def make_busy(x: str) -> int:
2      make_busy_list = []
3
4      idx = 0
5      while idx < len(x):
6          make_busy_list.insert(0, 'A')
7          idx = idx + (len(x) - idx) // 2 + 1
8
9      return 1
```

Question 3. [5 MARKS]

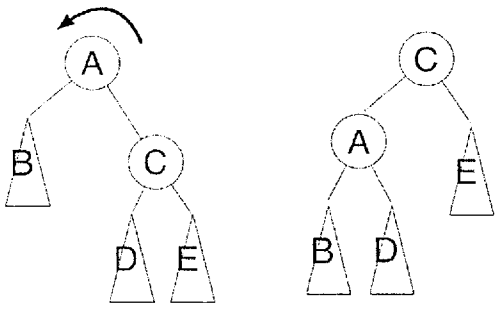
Part (a) [1 MARK]

Complete the is_heap function below according to its docstring. Suggestion: Start by writing two helper functions, one to check for each of the two properties that must hold for a tree to be a min-heap.

```
1  class BTNode(object):
2      def __init__(self, value, left, right):
3          self.value = value
4          self.left = left
5          self.right = right
6
7  def is_heap(root: 'BTNode') -> bool:
8      '''Return True iff the tree rooted at root is a binary min-heap.
9
10     Example:
11     >>> is_heap(None)
12     True
13     >>> is_heap(BTNode(
14                         0,
15                         BTNode(5, None, None),
16                         None
17                     ))
18     True
19     >>> is_heap(BTNode(
20                         0,
21                         None,
22                         BTNode(5, None, None)
23                     ))
24     False
25     '''
```

Part (b) [1 MARK]

Some kinds of trees require that nodes be rotated. Define a binary node class (or, if you don't, we'll assume you're using the one from part a) and then create a new `BinaryTree` class. The constructor for the class should accept one argument (in addition to the usual `self`) that is an instance of a binary tree node that will form the initial root of the tree (thereby freeing you from having to write any methods for inserting values into your rotating tree). In your `BinaryTree` class, write a `rotate_cc` method that rotates a node counterclockwise as in the diagram below. The method should accept one argument (in addition to the usual `self`) that is the binary node to be rotated.



Before rotation After rotation

Part (c) [1 MARK]

Complete the `is_node_in_cycle` function according to its docstring. All solutions longer than 25 lines (including helper functions and headers) will be given a 0.

```
1  class Node(object):
2      def __init__(self: 'Node') -> None:
3          self.edges = []
4
5      def set_edges(self: 'Node', edges: 'list of Nodes') -> None:
6          '''Connect Node self to the nodes in edges. Old edges are replaced.'''
7          self.edges = edges
8
9  def is_node_in_cycle(node: 'Node') -> bool:
10     '''Return True iff node is involved in a cycle.
11
12     Example:
13     >>> nodes = [Node(), Node(), Node()]
14     >>> nodes[0].set_edges([nodes[1]])
15     >>> nodes[1].set_edges([nodes[2]])
16     >>> nodes[2].set_edges([nodes[1]])
17     >>> is_in_cycle(nodes[0])
18     False
19     >>> is_in_cycle(nodes[2])
20     True
21     '''
```

Part (d) [1 MARK]

Complete the following function according to its docstring. The order of the items in the returned `list` does not matter.

```
1  def permutate(in_str: str) -> list:
2      '''Return a list of all permutations of the characters in in_str.
3
4      Examples:
5      >>> permutate('')
6      []
7      >>> permutate('ABC')
8      ['ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA']
9      >>> permutate('AA')
10     ['AA', 'AA']
11     '''
```

Part (e) [1 MARK]

Describe a set of steps for turning any while-loop into a recursive function.

This space intentionally left blank for extra space for answering questions, rough work, and drawings of the secret lives of test-case writers.

This space intentionally left blank for extra space for answering questions, rough work, and drawings of turtles.

Last Name: _____ **First Name:** _____

Bonus question [1 mark]

Give a point-estimate (as opposed to an estimated range) of your quiz mark out of 50 (excluding all bonuses) accurately and write your answer here:

This space intentionally left blank for extra space for answering questions, rough work, and drawings of what you will do with your newfound computer scientry powers (mad or otherwise). Remember, with great power comes great power.