First Name:
Last Name:
Student #:

NAME (PRINT): **Last/Surname** **First /Given Name**

STUDENT #: **SIGNATURE:**

UNIVERSITY OF TORONTO MISSISSAUGA
APRIL 2015 FINAL EXAMINATION
CSC148H5S
Introduction to Computer Science
A.Attarwala and T.Tong
Duration - 3 hours
Aids: None

The University of Toronto Mississauga and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for possessing any unauthorized aids during the writing of an exam. Clear, sealable, plastic bags have been provided for all electronic devices with storage, including but not limited to: cell phones, tablets, laptops, calculators, and MP3 players. Please turn off all devices, seal them in the bag provided, and place the bag under your desk for the duration of the examination. You will not be able to touch the bag or its contents until the exam is over.

If, during an exam, any of these items are found on your person or in the area of your desk other than in the clear, sealable, plastic bag; you may be charged with an academic offence. A typical penalty for an academic offence may cause you to fail the course.

*Please note, you **CANNOT** petition to **re-write** an examination once the exam has begun.*

First Name:
Last Name:
Student #:

You must obtain a mark of at least 40% on this exam, otherwise, a course grade of no higher than 47% will be assigned.

Name:
Student #:

Circle your instructor:

A. Attarwala

T. Tong

Question	Mark
1. Object Oriented	/14
2. Complexity	/12
3. Stacks and Queues	/15
4. Trees	/22
5. Sorting	/13
6. Singly Linked List	/24
Total:	/100

First Name:
Last Name:
Student #:

Question1) Object Oriented: **[14]**

a) Define *inheritance* (IS-A) and give one example. [2]

b) Define *composition* (HAS-A) and give one example. [2]

c) [10]
A fast food restaurant owner has asked you to design a system for them that lets them keep track of their orders in a computer. The following is the description they have given you:

When a customer places an order, they specify which item(s) from the menu they want and how many of each they would like. They also specify whether they want to upgrade any of the items to a combo, and whether they want to dine in or take out. The cashier should be able to input this information and create a new order. They should also be able to compute an order subtotal, and compute taxes by specifying the tax rate as a percentage. There are two menus: a main menu and a side menu and so there are two kinds of menu items. Only main menu items are upgradable to combo. Each menu item has a name, and a price. Side menu items come in three different sizes: small, medium and large.

Your task is to perform an object oriented analysis for this project, i.e. give a list of classes that are needed along with their *instance variables* and *instance methods.* You do not have to implement any of the instance methods, but just list them inside their respective classes (listing their signature and docstrings ONLY). If applicable, specify clearly how one class is related to the other via *inheritance* or *composition.*

First Name:
Last Name:
Student #:

Solution to Question 1.c):

First Name:
Last Name:
Student #:

Question 2) Complexity: **[12]**

The fibonacci series is defined as follows $f_0, f_1, f_2 \ldots f_n$

where $f_0=0$
$f_1=1$
$f_n=f_{n-1}+f_{n-2}$ (for n > 1)

a) Write a recursive function that returns back the nth fibonacci number i.e. f_n [1]

```
def nfibonacciNumber(n):
    '''given int n, return back fn where
    fn is the nth fibonacci number.Your solution MUST be recursive'''
```

b) Trace out all the recursive steps for the above function call `nfibonacciNumber(6)` and clearly state what the return value is? [3]

c) What is the BigO running time AND the BigO memory consumption of the above recursive solution? In addition to the correct answer, you MUST explain all your steps clearly. [4]

d) Now rewrite the function so that your solution is iterative. However your running time MUST be O(n) or better and that memory consumption is O(n) or better. [4]

```
def nfibonacciNumber(n):
    '''given int n, return back fn where
    fn is the nth fibonacci number. Your solution MUST be iterative'''
```

First Name:
Last Name:
Student #:

Question 3) Stacks and Queues: **[15]**

Implement a `Queue` ADT (Abstract Data Type) using two Stacks by completing the body of the methods below. This means that the elements inside your queue should be stored in the stacks instead of normal Python lists or linked lists (as you did in lecture). This question is continued on the next page. You can assume you have a `Stack` class (where the constructor creates an empty `Stack`) with `push`, `pop` and `is_empty` methods. You are not required to know the implementation detail of the `Stack` class to answer this question.

Hint: As scrap work, draw yourself two stacks and try *pushing* and *popping* elements to see how you can generate a `Queue` like behaviour.

Part a) [3]

```
from stack import Stack
class Queue(object):
    def __init__(self):
    '''initialize an empty queue'''
```

Part b) [2]

```
def enqueue(self,item):
'''add the given item to the end of the queue'''
```

First Name:
Last Name:
Student #:

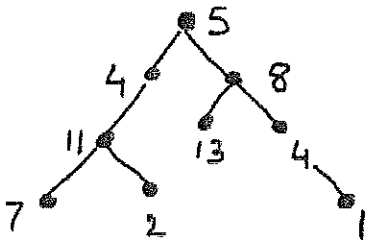
Part c) [10]

```
def dequeue(self):
    '''Remove and return the next item at the beginning/head of the queue'''
```

First Name:
Last Name:
Student #:

Question 4) Trees: **[22]**

a) Consider the following binary tree: [10]



The path from the 5 to the 7 includes the values 5,4,11, and 7, whose sum is 27.
The path from the 5 to the 2 includes the values 5,4,11 and 2, whose sum is 22.
The path from the 5 to the 13 includes the values 5,8 and 13, whose sum is 26.
The path from the 5 to the 1 includes the values 5,8,4, and 1, whose sum is 18.

Write the following function recursively. No helper functions are allowed. You can safely assume that `t` is a binary tree using the nodes and references version.

Hint: You need to recurse on the left and the right subtrees, what will you pass into these calls for the total parameter?

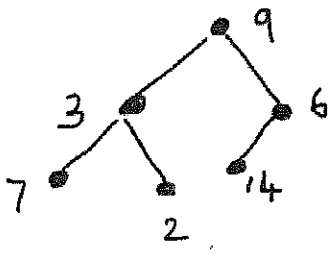
```
def has_path_sum(t,total):
    '''Given binary tree, t return a boolean value
       indicating whether there is a path in t to any
       leaf whose values add up to int total. Precondition: t
       is not None. total is integer.'''
```

First Name:
Last Name:
Student #:

b) Consider the function `foo` defined below: [12]

```
def foo(tree):
    if tree:
        while tree.right !=None:
            tree=tree.right
        return tree.root
```

Part a): [2]
What will `foo` return if passed the root of the following binary tree?



Part b): [2]
If `tree` is the root of a binary search tree, describe in 1 or 2 english sentences what does `foo` do?

Part c): [2]
Draw a `tree` with 5 nodes for which `foo` would have the longest running time.

Part d): [2]
Draw a `tree` with 5 nodes for which `foo` would have the shortest running time.

Part e): [2]
Which of the following best describes the running time of `foo` on a complete binary tree with `n` nodes? (Circle one)

O(1) O(log n) O(n) O(nlogn) $O(n^2)$

Part f): [2]
Which of the following best describes the worst case running time of `foo` on tree with `n` nodes? (Circle one).

O(1) O(log n) O(n) O(nlogn) $O(n^2)$

First Name:
Last Name:
Student #:

Question 5) Sorting **[13]**

For each part of this question, assume you start with the list `L = [9,4,7,2,1,8]`. In order to get part marks, you should give all steps (circle the final result).

a) Consider the running `insertion sort` (ascending order) on `L`. Show what `L` looks like on each step (one iteration of the outer loop) before it is completely sorted. [4]

b) Consider running `selection sort` (ascending order) on `L`. Show what `L` looks like on each step (one iteration of the outer loop) before it is completely sorted. [4]

c) Consider running `merge sort` (ascending order) on `L`. What does `L` look like on each step before it is completely sorted? Clearly show the `split` and then the `merge` on `L` before it is completely sorted. [5]

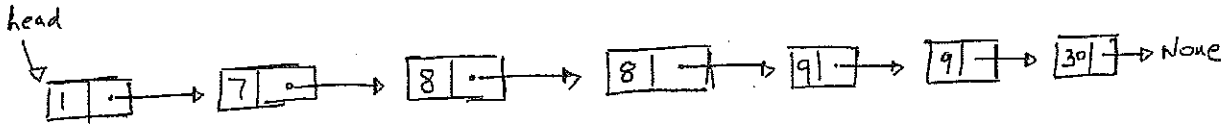
First Name:
Last Name:
Student #:

Question 6) Singly Linked List. **[24]**

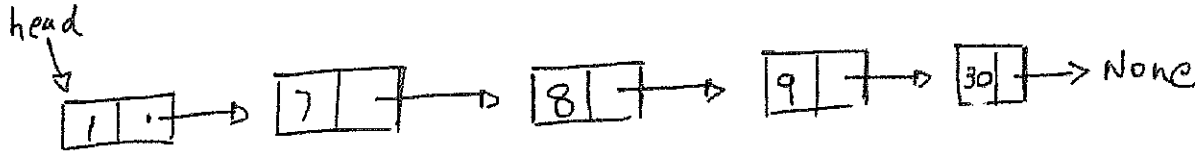
a) Write a function called `removeDuplicates(head)` that takes in a singly linked list in increasing order and deletes any duplicate nodes from the singly linked list. You can assume that the linked list passed in will ALWAYS be in increasing order. [12]

- You can only traverse the list once
- Your function *MUST BE NON-RECURSIVE.*

As an example if the following linked list is inputted in your function:



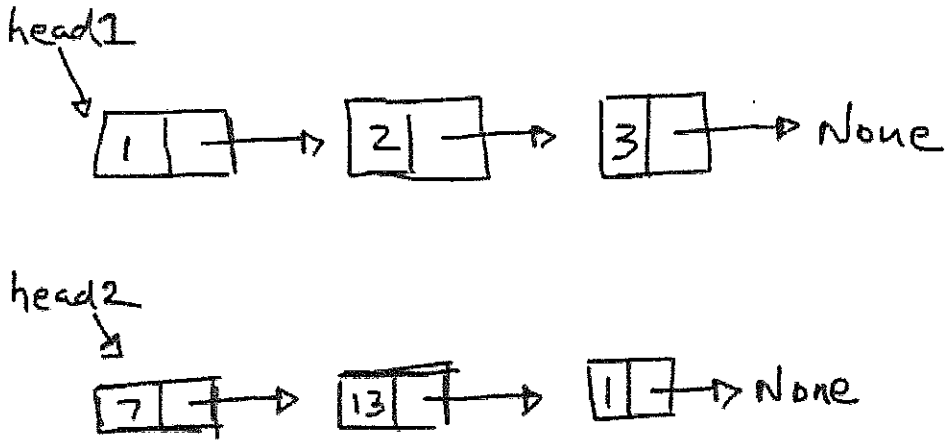
Your function MUST return the `head` of the same linked list with the duplicate nodes removed:



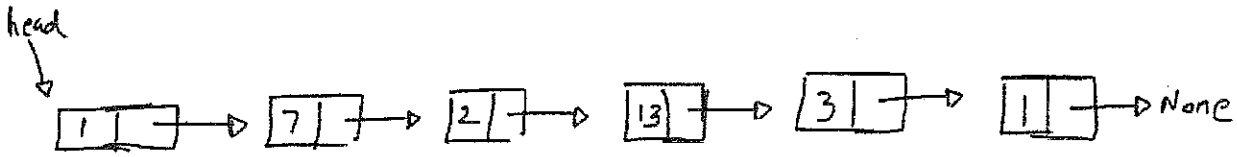

```
def removeDuplicates(head):
    '''head is the head of the singly linked list. This function returns back
       the head of same linked list with duplicate nodes removed'''
```

First Name:
Last Name:
Student #:

b) Given two singly linked lists, merge their nodes together to make one linked list taking nodes alternately between the two linked lists. Write a method `ShuffleMerge` that takes in as inputs two linked list referred to by `head1` and `head2` as in the example below: [12]



and returns the `head` of a merge singly linked list.



If either of the singly linked list runs out of nodes, all the nodes should be taken from the other linked list and merged at the end. Write your code on the next page. Your function *MUST BE NON-RECURSIVE.*

First Name:
Last Name:
Student #:

```
def shuffleMerge(head1,head2):
    '''merge the nodes of the two singly linked lists into a single list
    taking a node alternately from each list starting from head1. This function will
return the head of the merged list.'''
```

First Name:
Last Name:
Student #:

Appendix:

For Question 4

```
class BinaryTree:
 '''You can safely assume that this class has been
    correctly and fully implemented as per the above definition of Binary Tree.'''

    def __init__(self,rootValue):
        '''Creates a new BinaryTree where, rootValue refers to the integer value at
root          node. The left child is set to None, and the right child is set to
None'''
        self.root=rootValue
        self.left=None
        self.right=None
    .
    .
```

From the above code of BinaryTree:

`t.root` refers to the integer value at the root node of the tree `t`.
`t.left` is a reference to the left subtree of tree `t`.
`t.right` is a reference to the right subtree of tree `t`.

You can directly refer to the instance variables of the binary search tree when writing your code.

==========

For Question 6:

```
class Node:
    def __init__(self, e,n=None):
        '''e is the integer that will reside in the node.
        n is the reference to the next node'''
        self.element = e
        self.next = n
    def __str__(self):
        return str(self.element)
```