First Name:
Last Name:
Student #:

CSC 148 Summer 2015
Term Test
10th June 2015

University of Toronto
Mississauga

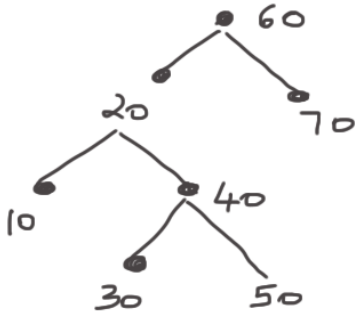
In addition to the correct answer, you MUST show all your work in order to receive full credit.

Questions	Mark:
Question1) Multiple Choice Questions	/10
Question 2) Binary Trees	/15
Question 3) Linked List	/20
Total:	/45

THIS EXAM CONTAINS A TOTAL OF 10 PAGES. PAGE 9/10 IS THE APPENDIX OF YOUR EXAM

Question 1: Multiple Choice Questions **[10]**

1) Refer to the following tree when answering 1.a) and 1.b) of the multiple choice question.



1.a) What would the result of a *pre-order* traversal of the above tree? [2]

a) 60 20 10 40 30 50 70
b)10 20 30 40 50 60 70
c)10 30 50 40 20 70 60
d) 70 60 50 40 30 20 10
e) None of the above

Explain your answer (in 1 or 2 sentences):

1.b) What would the result of a *post-order* traversal of the above tree? [2]

a) 60 20 10 40 30 50 70
b)10 20 30 40 50 60 70
c)10 30 50 40 20 70 60
d) 70 60 50 40 30 20 10
e) None of the above

Explain your answer (in 1 or 2 sentences):

2) How many times is the string "`foo`" printed by the function call `foo(4)` ? [3]

```
def foo(i):
    if i>2:
        foo(i/2)
        foo(i/2)
       foo(i/2)
    print ("foo")
```

a) 3
b) 4
c) 7
d) 8
e) Something else

Explain your answer:

3) What is the return value for the function call `mysteryFunction([1,3,5,2,4,6])` [3]

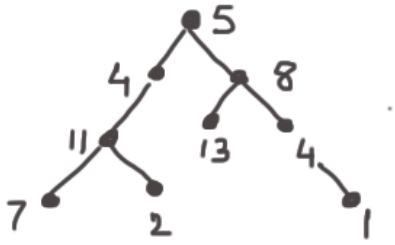
```
def mysteryFunction(x):
    if not x:
        return []
    if x[0] % 2 == 0: #even
        return [x[0]] + mysteryFunction(x[1:])
    else:
        return mysteryFunction(x[1:]) + [x[0]]
```

a) The function has no base case and will result in stack overflow
b) Something else
c) [1,2,3,4,5,6]
d) [1,3,5,2,4,6]
e) [1,3,5,2,6,4]
f) [1,3,5,4,6,2]
g) [2,4,6,1,3,5]
h) [2,4,6,5,3,1]

Explain your answer:

Question 2) Binary Trees (See Appendix for this question) **[15]**

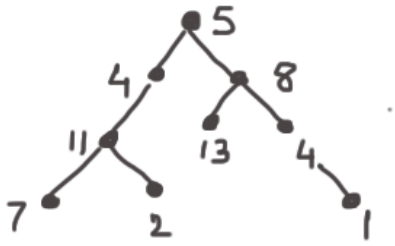
a) Consider the following binary tree: [5]



This binary tree has a total of 9 nodes. Write a *recursive* Python function such that it takes in as input a binary tree and returns back an integer that represents the count of all nodes in the binary tree. No helper functions are allowed. YOU MUST CLEARLY MARK YOUR BASE CASE(S) AND YOUR RECURSIVE CASE(S). You can safely assume that `t` is a binary tree using the nodes and references version.

```
def count_nodes_in_binaryTree(t):
    '''Given binary tree, t return an integer that represents the total count of all
nodes in binary tree'''
```

b) 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 using *recursion*. No helper functions are allowed. YOU MUST CLEARLY MARK YOUR BASE CASE(S) AND YOUR RECURSIVE CASE(S). 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?

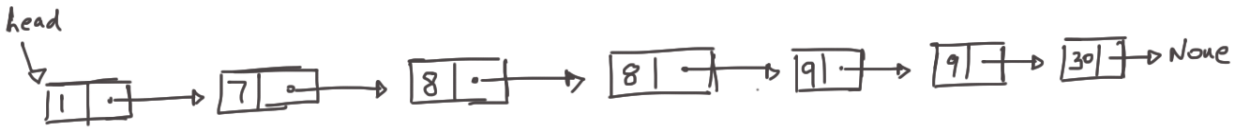
```
def has_path_sum(t,total):
    '''Given binary tree, t return a boolean value
       indicating whether there is a path in t to any
       leaf whose values add up to integer total.'''
```

Question 3) Singly Linked List. **(See Appendix for this question)** **[20]**

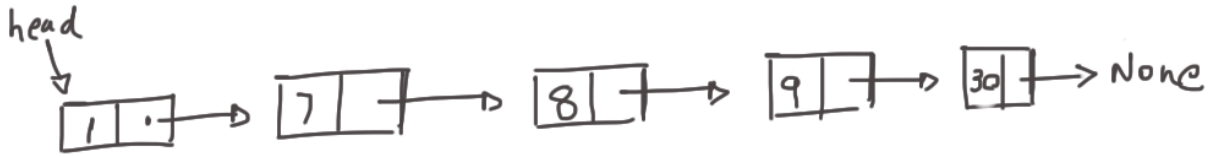
a) Write a function called `removeDuplicates(head)` that takes in a singly linked list in increasing order and deletes any duplicate nodes from the singly linked list. You can assume that the linked list passed in will ALWAYS be in increasing order. [12]

- 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:

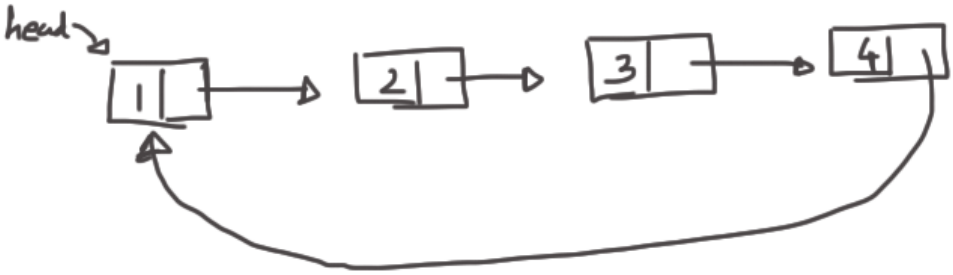


Your function MUST return the `head` of the same linked list with the duplicate nodes removed:




```
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'''
```

b) A circular linked list is a linked list where the `next` of the last node refers back to the `head` node. Write a recursive algorithm in Python that determines if the linked list is circular or not. You MUST write two functions to implement this algorithm. The function `is_circular` takes in as input the head of a singly linked list and returns back `True` if the linked list is circular, otherwise returns `False`. The helper method `__is_circular` is recursive function and MUST be called from the method `is_circular`. YOU MUST DECIDE CAREFULLY WHAT THE FUNCTIONAL TYPE OF THE HELPER METHOD is. Following is an example of a linked list that is circular in nature. You *do not* have to consider input case(s) of linked list that have circular loops not ending at the head node. [8]




```
def is_circular(head):
    '''head is the head of the linked list. This function returns back
       True if the linked list is circular or returns back False is not. The
       type of head is Node'''
```

```
def __is_circular(                                                    ):
    '''THIS IS A RECURSIVE FUNCTION. THE FUNCTIONAL TYPE OF THIS FUNCTION MUST
       BE DECIDED BY YOU'''
```

Appendix:

For Question 2

```
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`.

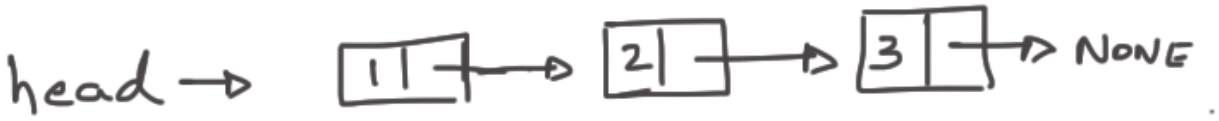
You can directly refer to the instance variables of the binary search tree when writing your code.

For Question3

```
class Node:
    '''Node is a class that represents a single Node in a Linked List'''
     def __init__(self,element,next=None):
       '''Creates a Node instance. The _element of this instance is set to element and
       the _next of this instance is set to next'''
          self._element=element
          self._next=next
```

An example of creating a linked list using the Node class is as follows:

```
head=Node(1,Node(2,Node(3)))
```



Extra WorkSheet

THIS SHEET WILL ONLY BE MARKED BY US IF YOU HAVE YOUR NAME AND STUDENT # AT THE VERY TOP

Extra WorkSheet

THIS SHEET WILL ONLY BE MARKED BY US IF YOU HAVE YOUR NAME AND STUDENT # AT THE VERY TOP