School of Information Systems

Past Year Paper: AY 2015-16 Term 1 Quiz 1 and 2 (Combined)

Course	**COR–IS1702 Computational Thinking**
Name of Student	VeryEvil CuteBunny
Group (circle one)	G1, G2, G3, G4, G5, G6, G7, G8

INSTRUCTIONS TO CANDIDATES

1. Please do not turn over this cover page until you are told to do so.

2. This examination paper contains **8** questions, and comprises **13** pages including this instruction sheet.

3 The time allowed for this mock examination paper is **90 minutes**. Within this time, you are expected to attempt **ALL** questions.

4 You are not allowed to use a **calculator**.

5 Write all answers on this paper. Additional blank sheets will be provided upon request.

6 The questions require short answers. Long-winded explanation or illegible handwriting will be penalized.

	Topic	Score
Q1	Counting (5 marks)	
Q2	Complexity (5 marks)	
Q3	Searching (5 marks)	
Q4	Sorting (5 marks)	
Q5	Stack, Queue (3 marks)	
Q6	Recursion (7 marks)	
Q7	Binary Tree (5 marks)	
Q8	Graph (5 marks)	Note for 2020/21 Term 1 students: "Graph" is out of scope for the mid-term exam

1. (Total: 5 marks) *Counting*

There are 10 men and 10 women.

(a) How many ways are there to arrange these people in a row? (2 marks)

(b) How many ways are there to seat these 20 people, if there are two different circular tables? One table is blue, and the other table is red. Each table has 5 men and 5 women alternating. Two seatings for a table are considered the same when everyone on that table has the same two neighbors without regard to whether they are right or left neighbors. (3 marks)

2. (Total: 5 marks) *Complexity*

It is given that `submethod` is a function that takes an integer n as input, and has a complexity of O(n).

(a) What is the Big O complexity of `method_2a` ? Clearly show your working. (2 marks)

Hint: $\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + (n-1) + n = \frac{n(n+1)}{2}$

```
01:  def method_2a(n):
02:      for i in range(1, n):
03:          for j in range(1, i):
04:              submethod(n)
```

(b) What is the Big O complexity of `method_2b` ? Clearly show your working. (1.5 marks)

Hint: $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 + n^2 = \frac{n(n+1)(2n+1)}{6}$

```
01:  def method_2b(n):
02:      for i in range(1, n):
03:          for j in range(1,(i*i)):
04:              submethod(n)
```

(c) What is the Big O complexity of `method_2c` ? Clearly show your working. (1.5 marks)

Hint: $\sum_{i=1}^{n} 1/i = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n-1} + \frac{1}{n} \approx O(\log n)$

```
01:  def method_2c(n):
02:      for i in range(1, n):
03:          for j in range(1, n):
04:              if j % i == 0:
05:                  for k in range(1, n):
06:                      submethod(n)
```

3. (Total: 5 marks) *Searching*

(a) Suppose that you run binary search on an array A (the algorithm is given below). What are the final values of lower and upper when you search for number 21? Clearly show how you got the results. Note that index starts from 0, i.e., A[0] = 3. (2 marks)

A = [3, 7, 10, 15, 21, 36, 44, 52, 58, 65, 79, 85, 87, 90, 93]

Binary search algorithm:

```
def bSearch(array, target):
    lower = -1
    upper = len(array)
    while not (lower + 1 == upper):      #fail if region empty
        mid = (lower + upper)//2         #find middle of region
        if target == array[mid]:         #succeed if k is at mid
            return mid
        elif target < array[mid]:
            upper = mid                  #search lower region
        else:
            lower = mid                  #search upper region
    return -1                            #not found
```

Answer:
You may wish to use the following table for your working.

lower	*upper*	*mid*

(b) You are given as input an unsorted array B, containing n distinct integers (no duplicates) in the range from 1 to (n+1), whereby exactly one integer in this range is missing from the array. The objective is to determine which integer is missing. For example, for the array B = [2, 3, 5, 6, 1, 7], we have n = 6, and the range is from 1 to 7. The missing integer is 4.

Provide an algorithm (in pseudo code or Python code) to find the missing integer in an input array B. The lower the complexity of your algorithm is, the higher will the marks be. (Maximum 3 marks)

4. (Total: 5 marks) *Sorting*

(a) Suppose a sorted array C = [5, 8, 12, 19, 22, 36, 40, 52, 58, 63] is the output you get after applying insertion sort algorithm on another input array C'. What is the input array C' that would require:

i. the fewest number of comparisons. What is that number of comparisons? (1 mark)

ii. the most number of comparisons. What is that number of comparisons? (1 mark)

~~**(b)** You are given K ascending sorted arrays $\{A_1, A_2, ..., A_K\}$, where each array A_i has n integers. For example, the input may be K = 4 sorted arrays of n = 3 integers each, i.e., A_1 = [1, 3, 5], A_2 = [4, 9, 10], A_3 = [2, 6, 7], and A_4 = [8, 12, 15]. The objective is to combine these K input arrays into a single sorted array. The expected output is the combined sorted array, i.e., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15].~~

~~i. Provide an algorithm (in pseudo code or Ruby code) to perform this task. (2 marks)~~

~~ii. What is the Big-O complexity of your proposed algorithm in terms of K and n? (1 mark)~~

Question 4(b) is already in your tutorial (see Week 4 (Iteration & Decomposition) tutorial)

5. (Total: 3 marks) ***Stack and Queue***

(a) What is the sequence of values printed out by the following operations? (1.5 marks)

```
1.    s1 = Stack()
2.    s2 = Stack()
3.    s1.push(7)
4.    s1.push(12)
5.    s2.push(s1.pop)
6.    s2.push(s1.pop)
7.    p s2.pop #print out the value popped from s2
8.    s1.push(8)
9.    s1.push(28)
10.   p s2.pop #print out the value popped from s2
11.   s2.push(s1.pop)
12.   s2.push(s1.pop)
13.   p s2.pop #print out the value popped from s2
14.   p s2.pop #print out the value popped from s2
```

(b) What is the sequence of values printed out by the following operations? (1.5 marks)

```
1.    q1 = Queue()
2.    q2 = Queue()
3.    q1.enqueue(7)
4.    q1.enqueue(26)
5.    q1.enqueue(8)
6.    q1.enqueue(28)
7.    q2.enqueue(q1.dequeue)
8.    q2.enqueue(q1.dequeue)
9.    q2.enqueue(q1.dequeue)
10.   p q1.dequeue #print out the value dequeued from q1
11.   q1 = q2
12.   q2 = Queue.new
13.   q2.enqueue(q1.dequeue)
14.   q2.enqueue(q1.dequeue)
15.   p q1.dequeue #print out the value dequeued from q1
16.   q1 = q2
17.   q2 = Queue.new
18.   q2.enqueue(q1.dequeue)
19.   p q1.dequeue #print out the value dequeued from q1
20.   q1 = q2
21.   q2 = Queue.new
22.   p q1.dequeue #print out the value dequeued from q1
```

6. (Total: 7 marks) ***Recursion***

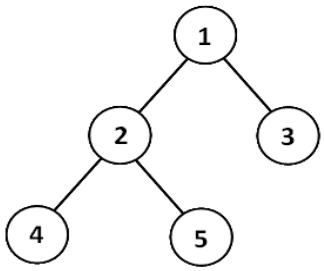
Consider the problem of getting the mirror of a given binary tree. The mirror of a binary tree is another binary tree with the left and right children of all non-leaf nodes interchanged. For example, the following figure shows two binary trees that are mirrors of each other.

Mirror Binary Trees

The following iterative function `mirrorIterativeQueue` is to get the mirror of an input binary tree rooted at `root`.

```
01:   def mirrorIterativeQueue(root):
02:       nodeQ = Queue()
03:       nodeQ.enqueue(root)
04:       while nodeQ.count() > 0:
05:           node = nodeQ.dequeue
06:           print(node.value) #print the value of the node
07:
08:           if(node.left != None):
09:               nodeQ.enqueue(node.left)
10:           if(node.right != None)
11:               nodeQ.enqueue(node.right)
12:
13:           temp = node.left
14:           node.setLeft(node.right) #set the left child of node
15:           node.setRight(temp) #set the right child of node
```

(a) What is the sequence in which the values of the nodes are printed (**line 6**) when running `mirrorIterativeQueue` with the following input binary tree A? (1 mark)



Binary Tree A

(b) Suppose that the complexity of each `enqueue` or `dequeue` operation is O(1). What is the overall Big O complexity of `mirrorIterativeQueue`? (1 mark)

(c) The function `mirrorIterativeQueue` above is implemented using a queue (**line 2**). Rewrite it into another function `mirrorIterativeStack` using a stack instead of a queue. The template is given below, and you may fill in the blank lines. (2 marks)

Answer:

```
def mirrorIterativeStack(root):
    nodeS = Stack()


    while nodeS.count() > 0:
```

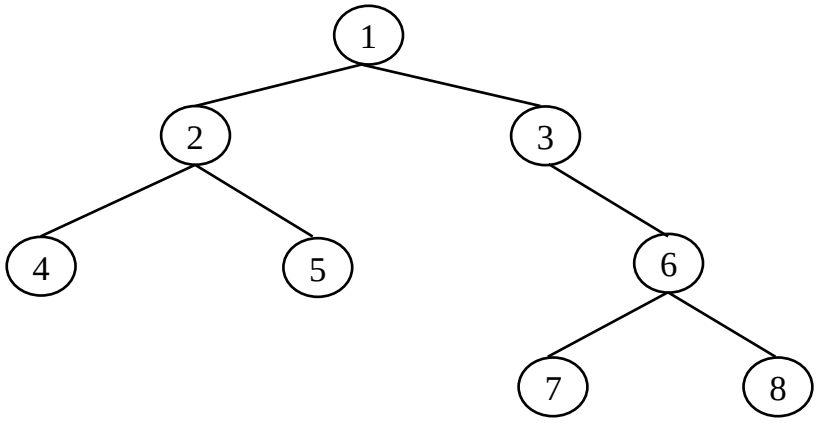
(d) Design an algorithm to derive the mirror of a binary tree using recursion.

- Provide a pseudocode for `mirrorRecursive` using recursion. (2 marks)

- What is the Big O complexity of this recursive function? (1 mark)

7. (Total: 5 marks) ***Binary Tree***

(a) Given a binary tree as follows:

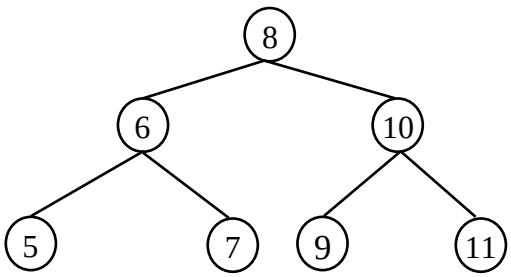


Which traversal method(s), out of: preorder, inorder, postorder, will visit 7 before 3 on the tree above? (1 mark)

(b) Given a list of integers: `15, 3, 12, 60, 72, 23, 10, 88, 20, 1, 65, 2`. Draw the binary search tree (BST) obtained by inserting these integers, in the sequence as listed above, into an initially empty BST (Hint: 15 is the root node). (2 marks)

(c) Given an array of distinct numbers, design an algorithm (in pseudocode or Python code) that will return `True` if the input array could conceivably be the postorder traversal of some binary search tree containing elements in that array, and return `False` otherwise.

For example, if the input is `a = [5, 7, 6, 9, 11, 10, 8]`, the algorithm should return `True`, since there exists a BST (shown below) for which `a` is the postorder traversal.

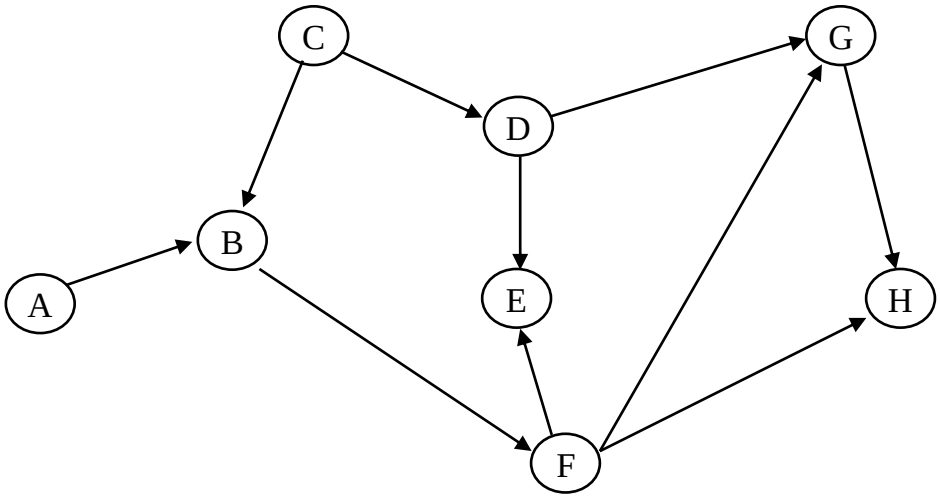


For another example, if the input is `b = [3, 1, 2]`, the algorithm should return `False`, since no BST containing these integers could possibly have `b` as its postorder traversal.

Your algorithm will be marked based on its complexity. Lower complexity translates to higher marks. (2 marks)

8. (Total: 5 marks) ***Graph***

Given a graph X as follows:



(a) Write down the adjacency list of the graph X above. The neighbours of each vertex are to be listed in alphabetical order. (1 mark)

Answer:

A	
B	
C	
D	
E	
F	
G	
H	

(b) What is the sequence of nodes visited by recursive depth-first-search when traversing the graph X, starting from **A**, based on the adjacency list above? (1 mark)

(c) What is the sequence of nodes visited by breadth-first-search on graph X, starting from vertex **C**, based on the adjacency list above? Clearly show your workings. You should enqueue each vertex's neighbors according to the sequence in the adjacency list. (1 mark)

Answer:

You may use the following table for your work:

Step	Dequeued	Visited/enqueued	Queue: [Head…Tail]
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

(d) Find two topological orderings in graph X. (2 marks)

~ End