SCHOOL OF COMPUTING

ASSESSMENT FOR
Special Term I, 2020/2021

CS1010X — PROGRAMMING METHODOLOGY

June 2021 Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. Please write your Student Number only. Do not write your name.
2. The assessment paper contains **SEVEN (7) questions** and comprises **THIRTEEN (13) pages**.
3. Weightage of questions is given in square brackets. The maximum attainable score is 48.
4. This is an **OPEN** book assessment, but you are NOT allowed to use any electronic devices such as laptop, mobile phone, and calculator.
5. Write all your answers in the space (enclosed in a box after each part/question) provided in this booklet.
6. You should make effort to write your answer legibly and neatly to avoid any possible confusion in interpretation. Use pencil where applicable (to plan and present your Python code with proper indentation) to ease any amendment along the way.

Student No: | | | | | | | | | |

(this portion is for the examiner's use only)

Question	Allocated	Your Marks
Q1		
Q2		
Q3		
Q4		
Q5		
Q6		
Q7		
Total		

Question 1: Warm up - fill in the blank [1 marks]

Dynamic programming can be viewed as a way to trade _____ for a better _____ complexity.

Question 2: Python Expressions [16 marks]

There are several parts to this problem. Answer each part **independently and separately**. In each part, one or more Python expressions are entered into the interpreter (Python shell). Determine all the response printed by the interpreter for the expressions entered and **write the exact output in the answer box**.

It is assumed that there is no syntax errors in our input to the Python interpreter. If the interpreter produces an execution error message at some stage, state where and the nature of the error (you don't need to show the exact error message but still need to write out any output from the Python expressions before the error was encountered). In another extreme, if the interpreter enters an infinite loop, explain how it happens. **Partial marks may be awarded for working if your answer is wrong. You are, however, responsible to make clear the part of your answer verse the part of your working.**

A. [4 marks]

```
def repeat_ops( op1, op2, start ):
    result = start
    count = 0
    while result > 1:
        count += 1
        if count%2:
            result = op1(result)
        else:
            result = op2(result)
    return count

print(repeat_ops(lambda x: x//2, lambda x:x+1, 128))
```

B. [4 marks]

```
def cross(a,b):
    if b==[]:
        return ()
    else:
        return ((a[0],b[0]),) + cross(a[1:], b[1:])

def accum(a):
    i = 1
    result = a.copy()
    while not(i > len(a)):
        cc = list(map(lambda x: x[0]+x[1], cross(result,result[i:])))
        result = result[:i] + cc
        i *= 2
    return result

print(accum([7,5,3,1,2,4,6,8]))
```

C. [4 marks]

```
t = (1,2)
q = ["a", "b"]
T = (11, t, q)
Q = [11, t, q]
t = (1,2,3)
q.append("c")
Q[2] = q + ["d"]
print(t)
print(q)
print(T)
print(Q)
```

D. [4 marks]

```
def CX (t1, t2, term):
    if (t1 == ()) or (t2 == ()):
        # base case 1: nothing to output
        return ()
    elif len(t1)==1 and len(t2)==1 and t1[0][0] != t2[0][0]:
        # base case 2: nothing to output
        return ()
    elif len(t1)==1 and len(t2)==1 and t1[0][0] == t2[0][0]:
        # base case 3: output a tuple derived from t1 and t2
        return (t1[0] + t2[0][1:] + term(t1[0], t2[0]),)
    elif len(t1) == 1 and len(t2) > 1:
        # recursive case 1
        return CX(t1, t2[0:1], term) + CX(t1, t2[1:], term)
    elif len(t1) > 1 or len(t2)==1:
        # recursive case 2
        return CX((t1[0],), t2, term) + CX( t1[1:], t2, term)
    else:
        print("impossible to reach here!")

t1 = ( (1, 10), (2, 20), (2, 21), (6, 60) )

t2 = ( (2, "B", 200), (1, "A", 100), (3, "C", 300),
       (6, "F", 600), (1, "Z", 101), (2, "Y", 201))

J = CX(t1, t2, lambda x, y: (x[1] + y[2],) )
for tup in J:
    print(tup)
```

Question 3: Iteration & Recursion, with Time & Space [13 marks]

Suppose you are given the following functions:

```
def C(n):
    if n<=1:
        return 0
    else:
        return n + C(n//2) + D(n-1)

def D(n):
    if n<=1:
        return 0
    else:
        return n + D(n//2) + C(n-1)
```

A. What is the output value of `C(8)`? [1 marks]

B. What is a good guess of the time complexity of `C` in terms of `n` where `n` is some positive number? You don't need to obtain the exact answer here as it may be complex, but your answer should be supported with good justification/reasoning to receive (partial) credit. [1 marks]

C. Provide an iterative implementation of `C` using a stack to simulate the given recursive version of `C`. We assume the input `n` is a positive integer, and you are not allowed to call any other functions (such as `D`). Please use only a simple list, but **NOT** tuple and **NOT** dictionary to implement your stack. Useful methods available for list are: `pop(i)` to remove the i^{th} item from a list, `append(item)` is to add `item` to the end of the list, and `len(`ℓ`)` to check the length of a list ℓ. All these mentioned methods take $O(1)$ time for a list. Please note that you are not asked to write the most efficient iterative implementation of the given function `C` in this part but just a working one with a stack. [4 marks]

```
def C_itr ( n ):
```

D. State and explain whether your above iterative implementation `C_itr` has a different time complexity from the given function `C`. No credit will be awarded if your `C_itr` is far from correct. [1 marks]

E. Provide a memoization implementation of `C` with an input `n` (a positive integer) without calling any other functions (such as `D`). We have fixed the use of a dictionary called `dic` here. Useful methods available for dictionary are: `get(n,-1)` to return the value of `C_memo(n)` if found else return -1, and `update({n:value})` to insert the `value` of `C_memo(n)` into a dictionary (or you can use the one given in our recitation `dic[n] = value`). We assume all these mentioned methods take $O(1)$ time for a dictionary. Note that no credit will be given if you simply copy the complex version of memoization in our lecture notes that keeps dictionary of dictionaries – for here, we expect your solution to use just the simple dictionary `dic`. [4 marks]

```
dic = {}
def C_memo ( n ):
```

F. State the time complexity of your `C_memo` in terms of `n`. Provide your justification too. No credit will be awarded if your `C_memo` is far from correct. [2 marks]

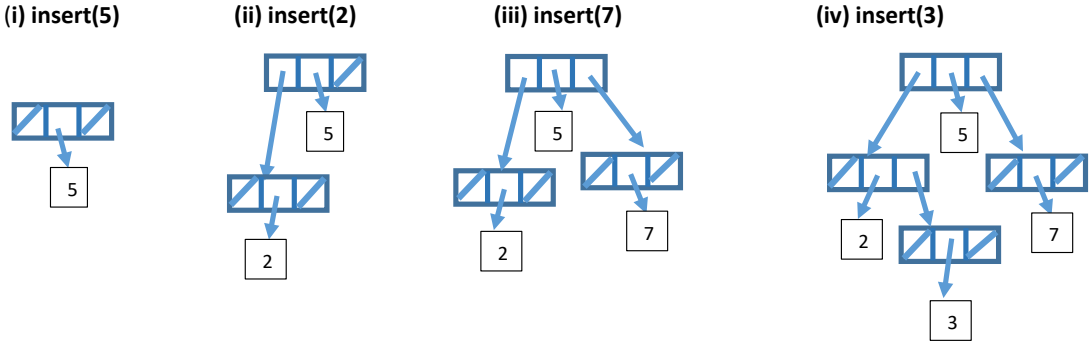
Question 4: The Mirror Image of a Tree [4 marks]

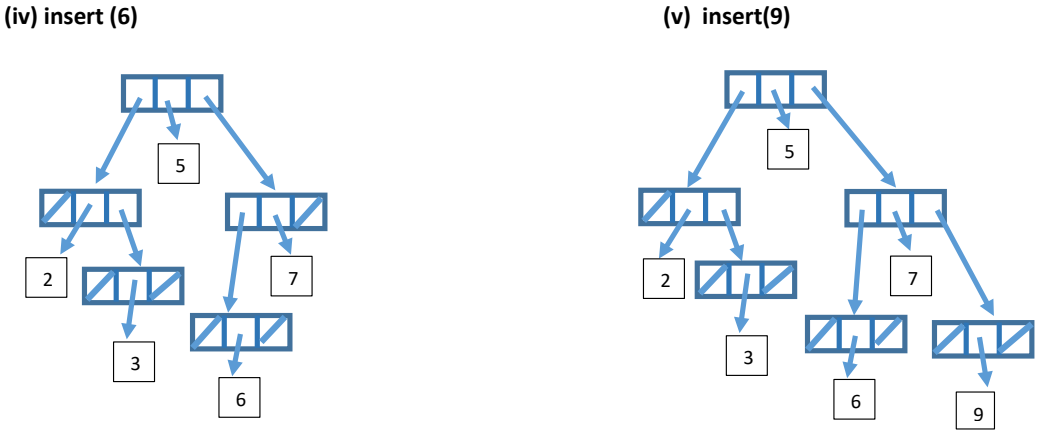
Recall from our Lecture 8 that a tree represented by a tuple is a tree where each element of the tuple is a child of the tree in the order of its appearance in the tuple. For a child that is a tuple itself, we have this child as a (smaller) tree (of the original tree) in a recursive manner. In this question, you are to write a function to compute the mirror image of a given tree. For example, the mirror image of a tree `((1,2,3), (4,5,6), (7,(8,9)))` is `(((9,8),7), (6,5,4), (3,2,1))`. That is, the mirror image has the children of the tree in the reverse order, and this is true recursively applied to each child.

```
def tree_mirror(tree):
```

Question 5: Binary Search Trees Deja Vu [4 marks]

We discussed binary search trees in Lecture 10 and you saw them in Mission 10 too. Below is the sequence of pictures when we insert numbers [5, 2, 7, 3, 6, 9] into the binary search tree (which originally is empty). Notice "/" in the picture means "None".

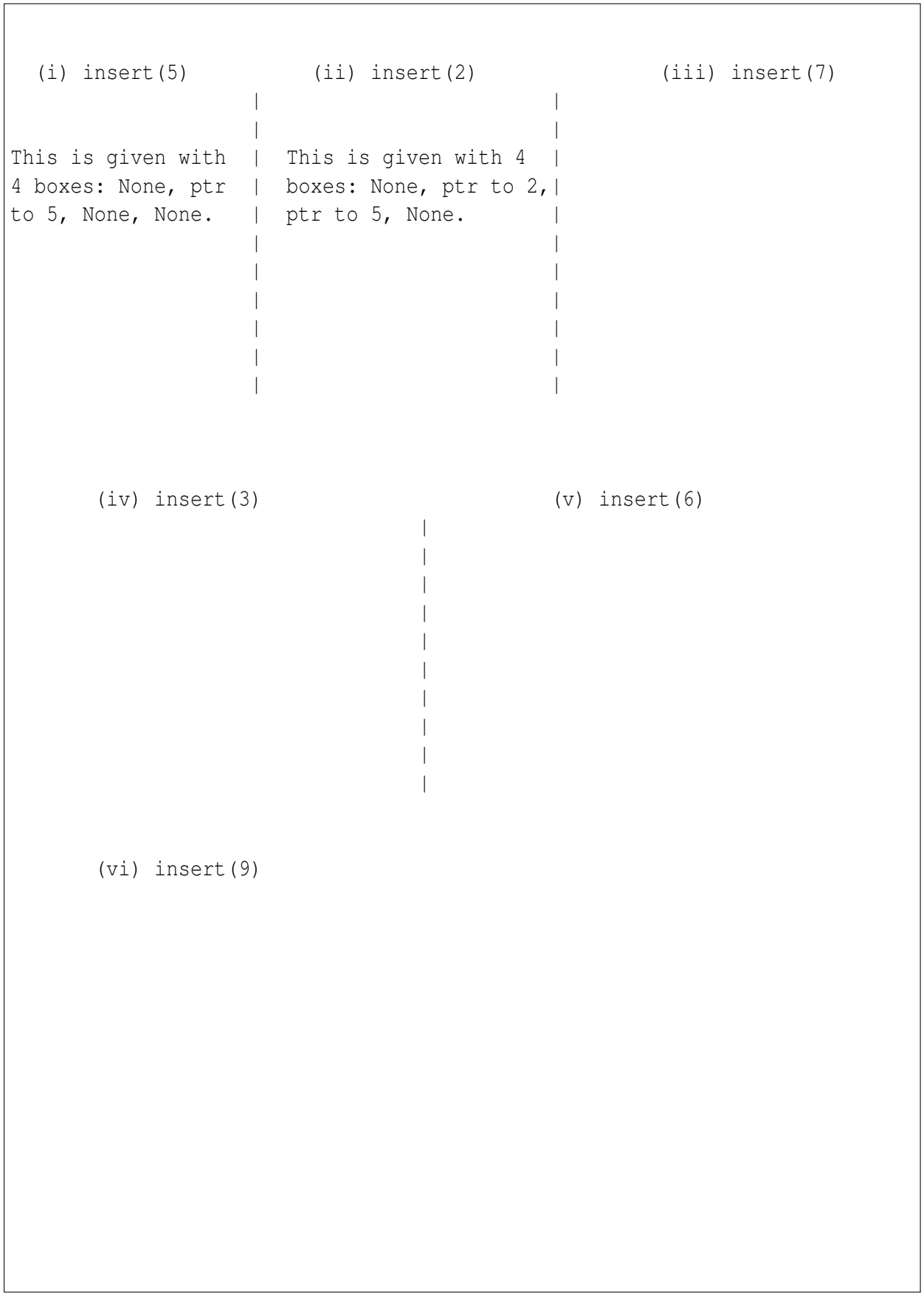




In this question, we study a variant to the binary search tree where each node is allowed to store up to 2 numbers (instead of just 1). That is, each node of the binary search tree has 4 elements: `left, entry1, entry2, right` (instead of `left, entry, right`) as shown in the above pictures). The following are true about this variant:

1. for a node, `entry1` cannot be `None`, while `entry2` can be `None`.
2. if `entry2` is not `None`, we have `entry1` $\leq$ `entry2`.
3. if a node has a `left` or `right` child, it must be the case that its `entry2` is not `None`.
4. the `left` sub-tree (if any) contains numbers that are no larger than `entry1`, while the `right` contains numbers (if any) that are no smaller than `entry2`.

With this understanding, you are to draw the sequence of pictures of how this variant of binary search tree looks like when you insert the sequence [5, 2, 7, 3, 6, 9]. You are to do this with the same style of drawing given in the above pictures in the next page. The first insertion of 5 and then the insertion of 2 are given to get you started.


(i) insert(5)
This is given with 4 boxes: None, ptr to 5, None, None.
(ii) insert(2)
This is given with 4 boxes: None, ptr to 2, ptr to 5, None.
(iii) insert(7)
(iv) insert(3)
(v) insert(6)
(vi) insert(9)

Question 6: Java – Object Oriented Concepts [6 marks]

Given the following Java classes, you are to list out those lines (between Line 25 to Line 30) in the main function that are not legal. That is, we want the line numbers of those statements that can result in compilation errors due to accessibility (visibility) consideration.

```
1    public class Base
2    {
3       public    int bPublic;
4       protected int bProtect;
5       private   int bPrivate;
6       // public methods follow (omitted here)
7    }
8
9    public class Derived extends Base
10   {
11       public  int dPublic;
12       private int dPrivate;
13       // public methods follow (omitted here)
14    }
15
16   public class Tester
17   {
18        private Base B = new Base();
19
20        public static void main(String[] args)
21        {
22            Base    b = new Base();
23            Derived d = new Derived();
24            System.out.println(
25                    B.bPublic + " " +
26                    b.bProtect + " " +
27                    d.bPrivate + " " +
28                    d.dPublic + " " +
29                    d.dPrivate + " " +
30                    d.bProtect
31                    );
32        }
33   }
```

Line _____ because _____

Line _____ because _____

Line _____ because _____

Question 7: Computational Thinking [4 marks]

Now that you have officially completed this course related to computational thinking. As a final assessment, you are to provide two good reactions you should have upon given the following Python program (with no syntax error and no other possible claim). That is, you are not asked to answer any specific question here, but given a chance to come up with questions about the given program that should be relevant to what you have learned in this course.

```
def find_separating_line( pts, pts_label ):

    # 2D points represented with [x,y,1]
    # eg. pts = [ [-6,-3,1], [-4,-5,1], [5,7,1], [3,5,1], [-4,7,1], [-1,3,1],
    #             [2,2,1], [5,2,1], [3,-3,1], [-2,1,1], [4,2,1] ]
    #
    # The corresponding labels of positive (+1) or negative (-1) for pts
    # e.g. pts_label = [+1, +1, -1, -1, -1, +1, +1, -1, +1, +1, -1]

    # Initialization
    W = [0,0,0]

    # looping to find the correct W
    while True:
        m = 0
        for i in range(len(pts)):
            if (W[0]*pts[i][0]+W[1]*pts[i][1]+W[2]*pts[i][2])*pts_label[i]<= 0:
                # update the new guess of W with the pts[i]
                W[0] += pts[i][0]*pts_label[i]
                W[1] += pts[i][1]*pts_label[i]
                W[2] += pts[i][2]*pts_label[i]
                m += 1

        if m == 0:
            print("Found:", W)
            return W
```

Here is a bit more about the above program. The input to the program is a collection of 2D points `pts` where each point has a corresponding positive (+1) or negative label (-1) in `pts_label`. In the example stated in the code, `pts[0]` is [-6,-3,1] with `pts_label[0]` of +1; `pts[1]` is [-4,-5,1] with `pts_label[1]` of +1; `pts[2]` is [5,7,1] with `pts_label[2]` of -1; and so on. Of course, the function is to accepted other possible inputs of `pts` and corresponding `pts_label`.

The program is an effort to find a line (represented by `W`) to separate the input points with positive label on one side of the line and negative label on the other side of the line. Technically, it means that the found `W` should be one such that the dot product of `W` and the `pts`, multiply with the `pts_label` is positive (see the `if` statement inside the `for` loop).

Note that you are **NOT** asked to trace the Python program carefully to then know how to answer this question.

— E N D O F P A P E R —