Easy4.0 CSC148

Mock Final Exam I

This Final Examination paper consists of 6questions with a sum of 76 points in total, and you must earn at least 40% to pass the exam. Comments and docstrings are not required except where indicated, although they may help us mark your answers.

- You do not need to put import statements in your answers.
- No error checking is required: assume all user input and all argument values are valid.
- If you use any space for rough work, indicate clearly what you want marked.

Name:________________

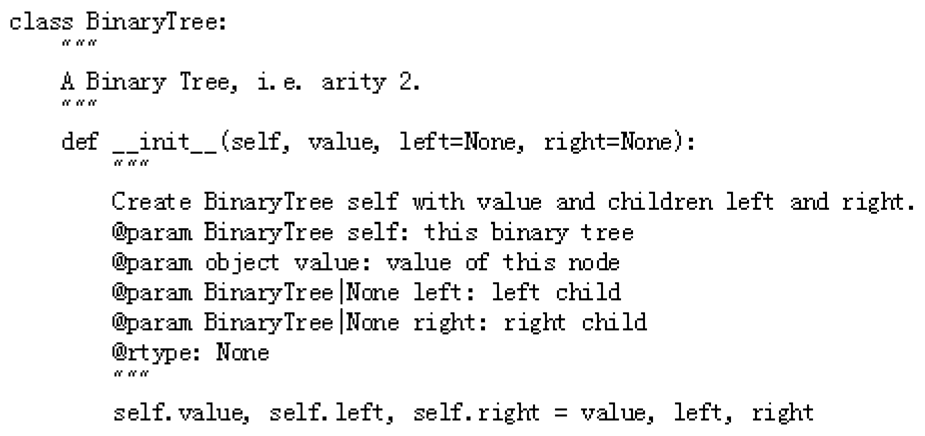
.

Marking Guide

Q1	**/12**
Q2	**/16**
Q3	**/8**
Q4	**/10**
Q5	**/18**
Q6	**/12**
Total	**/76**

Question 1 [12 Marks]

Recall the Binary Search Tree data structure we’ve defined in class.



```
class BinaryTree:
    """
    A Binary Tree, i.e. arity 2.
    """
    def __init__(self, value, left=None, right=None):
        """
        Create BinaryTree self with value and children left and right.
        @param BinaryTree self: this binary tree
        @param object value: value of this node
        @param BinaryTree|None left: left child
        @param BinaryTree|None right: right child
        @rtype: None
        """
        self.value, self.left, self.right = value, left, right
```

Part (a) [4 Marks]

Draw the **binary search tree** after insert 5 7 2 4 3 1 6 8 9, and Delete 7 and 5 from it, assume the binary search tree was initially empty. Assume we use the left subtree in deletion.

Part (b) [8 Marks]

```
def gather_less_than(bst,value):
    """ Return the list of values in the Binary Search Tree bst which are less than the
    value, in sorted order.
    @type bt: BinarySearchTree
    @type value: int
    @rtype: list of int

    >>> bt = BinarySearchTree(50)
    >>> bt.left = BinarySearchTree(30)
    >>> bt.right = BinarySearchTree(70)
    >>> gather_less_than(bt, 55)
    [30, 50]
    """
```

Question 2 [16 Marks]

Write Classes as given description below, the type contract is necessary, but all the descriptions are not necessary.

A Company needs to have internal structure design as follow.

There are different organizations, each organization has a name, parent organization and a level, the level would be either management or staff, other type input would cause an error called WrongOrgLevelException.

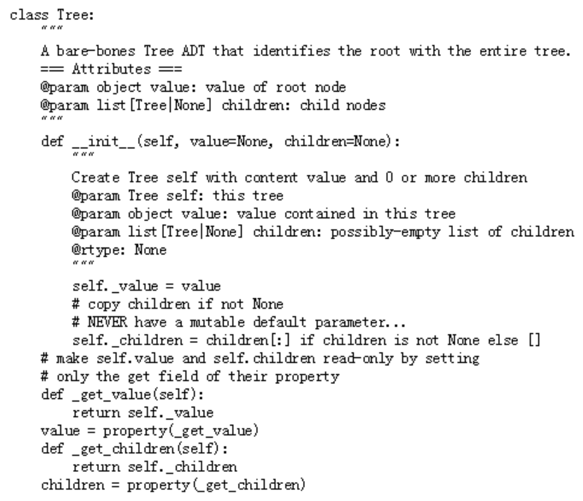
If no parent is indicated for the organization, its parent would be None. And we should have the ability to add employee to the organization and check if the employee is in the organization. NOTE: You don’t need to worry about deletion.

We have three different type of employees, each employee has a name, an employee id, an email address and a salary. Big Boss would have secretary which is another employee. Staff would have a specific desk number and the managers would have their own office location.

For the __init__ mehods, you should write both the docstring and the implementation, for other methods, you only need to write down the header and indicate the type contract.

Question 3 [8 Marks]

Recall the Tree data structure we’ve defined in class.



```
class Tree:
    """
    A bare-bones Tree ADT that identifies the root with the entire tree.
    === Attributes ===
    @param object value: value of root node
    @param list[Tree|None] children: child nodes
    """
    def __init__(self, value=None, children=None):
        """
        Create Tree self with content value and 0 or more children
        @param Tree self: this tree
        @param object value: value contained in this tree
        @param list[Tree|None] children: possibly-empty list of children
        @rtype: None
        """
        self._value = value
        # copy children if not None
        # NEVER have a mutable default parameter...
        self._children = children[:] if children is not None else []
    # make self.value and self.children read-only by setting
    # only the get field of their property
    def _get_value(self):
        return self._value
    value = property(_get_value)
    def _get_children(self):
        return self._children
    children = property(_get_children)
```

Implement the function “rotate_odd_level” defined below according to its docstring.

```
def rotate_odd_level (t: Tree) -> None:
    """ Rotate all odd level nodes, shift one to the left. For example, 1 2 3 would
        become 2 3 1. Assume the root is level 1, and its children are level 2.
    """
```

Question 4 [10 Marks]

```
class LinkedListNode:
    """
    Node to be used in linked list

    === Attributes ===
    next_ - successor to this LinkedListNode
    value - data represented by this LinkedListNode
    """
    next_: Union["LinkedListNode", None]

    def __init__(self, value: object,
                 next_: Union["LinkedListNode", None]=None) -> None:
        """
        Create LinkedListNode self with data value and successor next

        >>> LinkedListNode(5).value
        5
        >>> LinkedListNode(5).next_ is None
        True
        """
        self.value, self.next_= value, next_

class LinkedList:
    """
    Collection of LinkedListNodes

    === Attributes ==
    front - first node of this LinkedList
    back - last node of this LinkedList
    size - number of nodes in this LinkedList, >= 0
    """
    front: Union[LinkedListNode, None]
    back: Union[LinkedListNode, None]
    size: int

    def __init__(self) -> None:
        """
        Create an empty linked list.
        """
        self.front,self.back = None, None
        self.size = 0
```

Implement the method for linkedlist below.

```
def split_in_half(self: LinkedList) -> None:
    """ split each node in self into two nodes, each have the original item / 2,

    Pre-condtion: all nodes have number value.

    >>> m = LinkedList()
    >>> m.load_list([2, 4])
    >>> m.split_in_half()
    >>> print(m)
    1 -> 1 -> 2 -> 2 ->|
    """
```

Question 5 [18Marks]

Part (a) [6 Marks]

Provide a list that would make quicksort run in worst case, and use precise and clear English explain the reason.

Part (b) [6 Marks]

What would be the best case of inserting a new node into a binary search tree? And what would be the worst case. Indicate the time complexity of both and briefly explain using clear and precise English.

Part c [6 Marks]

What would be the time complexity of the previous question if we assume the tree is balanced? If you don’t think there would be cases, explain why.

Question 6 [12Marks]

What’s the runtime of the following function, please indicate the Big-O runtime with a brief explaination, pick the worst case if applicable.

```
def mystery1(n):
    if n < 100:
        for temp in range(n):
            print(‘Shui zui shuai?’)
    i = 0
    while i < n:
        print(‘Gary Shuai’)
        i += n // 10
```

```
def mystery2(n):
    if n == 1:
      return 1
   else:
     temp = 0
     for num in range(n):
       temp += num
     return temp + mystery2(n - 1)
```