Midterm 2

Graded

Student

Murali Karthik Ganji

Total Points

62 / 100 pts

Question 1

Short Answer **5** / 8 pts

1.1 **a** **1** / 1 pt

✔ **– 0 pts** Correct - True

– 1 pt Blank or incorrect

1.2 **b** **1** / 1 pt

✔ **– 0 pts** Correct - True

– 1 pt Blank or incorrect

1.3 **c** **1** / 1 pt

✔ **– 0 pts** Correct - False

– 1 pt Blank or incorrect

1.4 **d** **1** / 1 pt

✔ **– 0 pts** Correct - False

– 1 pt Blank or incorrect

1.5 **e** **0** / 1 pt

– 0 pts Correct - False

✔ **– 1 pt** Blank or incorrect

1.6 **f** **0** / 1 pt

– 0 pts Correct - Tail

✔ **– 1 pt** Blank or incorrect

1.7 **g** **0** / 1 pt

– 0 pts Correct - One

✔ **– 1 pt** Blank or incorrect

1.8 **h** **1** / 1 pt

✔ **– 0 pts** Correct - Aliases

– 1 pt Blank or incorrect

Question 2

Stacks and Queues **6** / 16 pts

2.1 **a - i** **0** / 2 pts

– **0 pts** Correct

✔ – **2 pts** Blank or incorrect

2.2 **a - ii** **2** / 2 pts

✔ – **0 pts** Correct

– **2 pts** Blank or incorrect

2.3 **b - i** **1** / 1 pt

✔ – **0 pts** Correct - Stack

– **1 pt** Blank or incorrect

2.4 **b - ii** **1** / 1 pt

✔ – **0 pts** Correct - Queue

– **1 pt** Blank or incorrect

2.5 **c** **0** / 6 pts

– **0 pts** Correct

– **2 pts** Did not correctly iterate over the string when adding to stack or did not correctly add to the stack

– **1 pt** Did not correctly check whether stack was empty for loop condition when removing from stack

– **1 pt** Did not pop off each character from the stack

– **1 pt** Did not build return string with the results of pop

– **1 pt** Does not return correctly or return correct value

– **0.5 pts** Minor syntax error

– **1 pt** Major or several minor syntax errors

– **5 pts** Did not use a stack

– **5 pts** Incorrect attempt

✔ – **6 pts** Blank or incorrect

2.6 d 2 / 4 pts

– 0 pts Correct

✔ – 1 pt `enqueue` doesn't insert at index 0

– 1 pt `dequeue` doesn't remove from the end of the list

✔ – 1 pt `dequeue`: does not return the item

– 2 pts `enqueue` blank or incorrect

– 2 pts `dequeue` blank or incorrect

– 0.5 pts Syntax error

Question 3

Recursion **16** / 20 pts

3.1 a **10** / 10 pts

- ✔ **– 0 pts** Correct
- **– 2 pts** Incorrect base case
- **– 2 pts** Does not slice list correctly in recursive call
- **– 2 pts** Does not recurse correctly
- **– 2 pts** Does not add 1 only when necessary
- **– 2 pts** Does not concatenate and return correctly
- **– 1 pt** Minor syntax error
- **– 2 pts** Major or several minor syntax errors
- **– 8 pts** Incorrect attempt
- **– 10 pts** completely incorrect or blank

3.2 b **6** / 10 pts

- **– 0 pts** Correct
- **– 2 pts** Sorted the list of indices
- **– 1 pt** Did not call the helper function correctly (if helper exists)
- **– 2 pts** Incorrect base case
- ✔ **– 2 pts** Incorrect recursive case
- ✔ **– 2 pts** Did not correctly concatenate recursive call and current element
- **– 1 pt** Did not return correctly
- **– 1 pt** Minor syntax error
- **– 2 pts** Major or several minor syntax errors
- **– 8 pts** Incorrect attempt
- **– 10 pts** Blank or incorrect
- **– 5 pts** Accesses incorrect elements in `str_list`
- **– 5 pts** Should not splice `str_list` on recursive call

1 you don't need pos and i, one parameter is enough.

Question 4

Linked Lists **6** / 20 pts

4.1 **a** **2** / 10 pts

– **0 pts** Correct

– **1 pt** Doesn't correctly handle an empty list

✔ – **4 pts** Does not correctly iterate through list

✔ – **4 pts** Does not correctly concatenate the values to a list to return

– **1 pt** Incorrect return

– **1 pt** Minor syntax error

– **2 pts** Major or several minor syntax errors

– **8 pts** Incorrect attempt

– **10 pts** Blank or incorrect

4.2 **b** **4** / 10 pts

– **0 pts** Correct

✔ – **2 pts** Does not handle deleting from index 0

Looping

– **1 pt** Does not correctly initialize cur and a variable to count the index

– **1 pt** Loop to iterate list is incorrect

– **1 pt** Does not update cur in loop

– **1 pt** Does not update variable to count the index in loop

---

Removing node

✔ – **1 pt** Does not store the correct node to delete in a variable

✔ – **1 pt** Does not change correct node's reference to skip deleted node

✔ – **1 pt** Does not remove deleted node's next reference

– **3 pts** Deleted rest of list by improperly updating next pointer

---

✔ – **1 pt** Does not correctly return the node

– **1 pt** Minor syntax error

– **2 pts** Major or several minor syntax errors

– **8 pts** Incorrect attempt

– **10 pts** Blank or incorrect

Question 5

Tree Basics **15** / 15 pts

5.1 **a** **2** / 2 pts

✔ **– 0 pts** Correct - No

– 2 pts Blank or incorrect

5.2 **b** **5** / 5 pts

✔ **– 0 pts** Correct

– 2 pts Only first 2 or 3 levels are correct

– 3 pts Incorrect attempt

– 5 pts Blank or incorrect

5.3 **c** **2** / 2 pts

✔ **– 0 pts** Correct - 4

– 2 pts Blank or incorrect

5.4 **d** **2** / 2 pts

✔ **– 0 pts** Correct - a, b, d, e, f, g, c

– 2 pts Blank or incorrect

5.5 **e** **2** / 2 pts

✔ **– 0 pts** Correct - d, f, g, e, b, c, a

– 2 pts Blank or incorrect

5.6 **f** **2** / 2 pts

✔ **– 0 pts** Correct - a, b, c, d, e, f, g

– 2 pts Blank or incorrect

Question 6

Binary Search Trees **9** / 16 pts

6.1 **a** **1** / 8 pts

- **– 0 pts** Correct
- **– 2 pts** Does not handle if tree is none
- **– 2 pts** Does not return node when found
- **– 1 pt** Returns newly created object instead of the actual node itself
- **– 1 pt** Returns the value instead of the node itself
- **– 2 pts** Does not recurse left when value is smaller
- **– 2 pts** Does not recurse right when value is larger
- **– 2 pts** Does not return recursive call
- **– 0.5 pts** Minor syntax or logic error
- **– 1 pt** Major or several minor syntax or logic errors
- ✔ **– 7 pts** Incorrect attempt
- **– 8 pts** Blank or incorrect

6.2 **b** **8** / 8 pts

- ✔ **– 0 pts** Correct
- **– 1 pt** Incorrect base case
- **– 2 pts** Does not handle if tree is none
- **– 2 pts** Does not handle if the tree is a leaf
- **– 4 pts** Does not add recursive calls together
- **– 4 pts** Incorrect recursive calls
- **– 4 pts** Counts all leaves
- **– 0.5 pts** Minor syntax error
- **– 1 pt** Major or several minor syntax errors
- **– 7 pts** Incorrect attempt
- **– 8 pts** Blank or incorrect

Question 7

Traversals to trees

5 / 5 pts

- ✔ **– 0 pts** Correct
- **– 2 pts** First 3 levels are correct
- **– 3 pts** First 2 levels are correct
- **– 4 pts** Incorrect attempt
- **– 5 pts** Blank or incorrect

Last name: Ganji First name: Murali Karthik

NetId: muralikarthik TA initial: MK

CSc 120: Introduction to Computer Programming II

Spring 2024

Midterm 2: Friday, April 5, 2024
Time: 50 minutes

In order to give all students the same amount of time to complete the exam, please do not open this exam until you are asked to do so. When told to begin, double-check that your name is at the top of this page and that your exam has all 11 pages.

IMPORTANT: You may not refer to any books, notes, or reference materials for this midterm.

Problem	Description	Earned	Max
1	Short answer		8
2	Stacks and queues		16
3	Recursion		20
4	Linked lists		20
5	Tree basics		15
6	Binary search trees		16
7	Traversals to trees		5
Total	**Total points**		**100**

1. Short Answer. [1 point each]

Note: Please write True or False for questions (a) through (e).

a) An abstract data type describes what the data represents, not how the data is represented.

True

b) A stack retrieves data in last in first out order.

True

c) A recursive function can be written without a base case.

False

d) In Python, the runtime stack is only used to handle recursive function calls.

False

e) It is not allowed to have multiple recursive calls inside a function.

True

Note: Answer (f) through (h) with one word.

f) What additional attribute (other than head) do we need to concatenate two linked lists without running a loop?

value

g) How many root nodes does a tree have?

0

h) If variables `x` and `y` refer to the same object, we say that `x` and `y` are what?

alias

2. Stacks and Queues. [16 points]

a) [4 points] Answer the following questions.

i) Stacks and queues are both linear data structures. What does it mean?

ii) What is the key *difference* between stacks and queues?

Stacks follows the LIFO (Last In First Out) method while Queue follows the FIFO (First In First Out) method.

b) [2 points] Specify whether a stack or queue would be the appropriate data structure for the problem below:

i) Check for Balanced Parentheses in an Expression: Stacks

ii) Assigning customers to tables in a restaurant: Queues

c) [6 points] Consider the following implementation of a `Stack` class:

```
class Stack:
    def __init__(self):
        self._items = []
    def push(self, item):
        self._items.append(item)
    def pop(self):
        return self._items.pop()
    def is_empty(self):
        return len(self._items)==0
```

Write a function `reverse(s)` that reverses the string `s` using a `Stack`. The function returns the reversed string.

d) [4 points] Implement the enqueue and dequeue operations of the following `Queue` class.
Requirements:

- Use the Python list provided below inside the constructor
- Make the front of the queue the *last* element of the Python list

```
class Queue:
    def __init__(self):
        self._items = []

    def enqueue(self, item):
        # write your code below
        self._items.append(item)
        return self._items.append(item)

    def dequeue(self):
        # write your code below
        self._items.remove(-1)
        self._items.pop()
```

3. **Recursion.** [20 points] For the problems below, you **must use recursion to receive full points**. Your solution may not have loop constructs or list comprehensions.

a) [10 points] Write a ***recursive*** function `ends_in_vowel(alist)` that takes a list of strings `alist` and returns a count of the strings in `alist` that end in a vowel. A vowel is defined as one of the characters in the string `"aeiou"`. For example, the call

```
ends_in_vowel(["fox", "lake", "area", "calm", "bee", "buzz"])
```

should return `3`. Assume that the strings in `alist` are all lowercase letters.

Note: You may **not** use a helper function.

```
def ends_in_vowel(alist):
    if alist == []:
        return 0
    else:
        ~~if alist[0][-1] in~~
        if alist[0][-1] in "aeiou":
            return 1 + ends_in_vowel(alist[1:])
        else:
            return ends_in_vowel(alist[1:])
```

b) [10 points] Write a *recursive* function `get_sublist(str_list, ind_list)` that takes a list of strings `str_list` and a list of indices `ind_list`, and returns a list containing strings of `str_list` at indices at `ind_list`. For example, the call

```
get_sublist(['Recursion', 'is', 'fun.', 'Sometimes', 'it',
'can', 'be', 'tricky.'], [0,1,7])
```

returns the list `['Recursion', 'is', 'tricky.']`.

Notes: You may assume that all of the integers in `ind_list` are valid indices of `str_list`. (I.e., no element of `ind_list` will cause an index out-of-bounds error.) The index list (e.g. `[0, 1, 7]`) is not necessarily sorted. It can contain duplicates as well.
Note: You **may** use a helper function.

```
def get_sublist(str_list, ind_list):
    return get_sublist_helper(str_list, ind_list, 0, 0)

def get_sublist_helper(str_list, ind_list, pos, i):
    if str_list == [] or ind_list == []:
        return []
    else:
        if ind_list[i] == pos:
            return str_list[pos] + get_sublist_helper(str_list,
                                   ind_list, pos+1, i+1)
        else:
            return get_sublist_helper(str_list, ind_list, pos+1, i)
```

4. **Linked Lists.** [20 points] For problems a) and b) below, use the following implementations of `Node` and `LinkedList`.

```
class LinkedList:
    def __init__(self):
        self._head = None

    def add(self,new):
        new._next = self._head
        self._head = new
```

```
class Node:
    def __init__(self,value):
        self._value = value
        self._next = None
```

Note: You may access the attributes directly without getter and setter methods. You may not change the implementation of `LinkedList` and `Node`.

a) [10 points] Write a **method** `listof_odds(self)` that takes the linked list argument `self` and returns a Python list of the node **values** that are odd. For example, if the linked list consists of the following node values (in this order):

`61, 4, 3, 7, and 12` .

then `listof_odds(self)` should return the **Python** list `[61, 3, 7]`. You may assume that all of the `_value` attributes of the nodes are non-negative integers.
Note: Use an iterative solution.

```
def listof_odds(self):
    current = self._head
    position = 0
    if current % 2 != 0:
        new._next = current
        current = new
    current = current._next
```

b) [10 points] Write a **method** `delete_from_pos(self, pos)` for the `LinkedList` class that deletes the node at position `pos` and returns the deleted node. Node positions begin at 0, i.e., the first node in the list has position 0. You can assume that `pos` >= 0 and `pos` < length of the linked list.
Note: Use an iterative method.

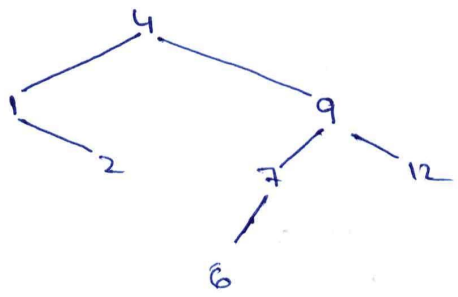
```
def delete_from_pos(self, pos):
    current = self._head
    position = 0

    while position < pos:
        current = current._next
        position += 1

    current = current._next
```

5. **Tree Basics.** [15 points]

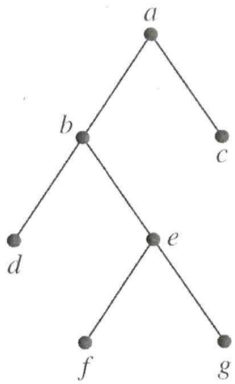
a) [2 points] Can a leaf node in a tree have children? Answer yes or no.

No

b) [5 points] Consider the integer list 4, 9, 1, 2, 7, 12, 6 inserted in a **Binary Search Tree** according to the order provided from left to right. Draw the final tree.



Use the tree below to answer the following questions:



c) [2 points] How many leaf nodes does the tree have?

4

d) [2 points] Write the preorder traversal:

[a, b, d, e, f, g, c]

e) [2 points] Write the postorder traversal:

[d, f, g, e, b, c, a]

f) [2 points] Write the breadth first traversal:

[a, b, c, d, e, f, g]

6. **Binary Search Trees.** [16 points]
For the following problems use the implementation of a tree Node below:

```
class BinaryTree:
    def __init__(self, value):
        self._value = value
        self._left = None
        self._right = None
```

Also, assume that `value()`, `left()`, and `right()` are the usual getters for the attributes.

a) [8 points] Write a **function** `search(tree, val)` that takes a binary search tree `tree` and a value `val`, and returns the node that has a value equal to `val`. If there is no such node, the method returns `None`. You **must** take advantage of the key property of a Binary search tree for full credit.

```
def search(tree, val):
    if tree is None:
        return 0
    elif tree._value == val:
        return val + search(tree._left, val) + search(tree._right, val)
    else:
        return search(tree._left, val) + search(tree._right, val)
```

b) [8 points] Write a **function** `count_leaves(tree)` that returns the count of the leaf nodes in the binary tree `tree`.

```
def count_leaves(tree):
    if tree is None:
        return 0
    elif tree._left is None and tree._right is None:
        return 1 + count_leaves(tree._left) + count_leaves(tree._right)
    else:
        return count_leaves(tree._left) + count_leaves(tree._right)
```

7. **Traversals to trees.** [5 points] Given the preorder and inorder traversals below, draw the resulting tree.

Preorder: 7, 12, 4, 9, 30, 13, 2, 27
Inorder: 4, 12, 9, 7, 2, 13, 30, 27