INDIAN INSTITUTE OF TECHNOLOGY ROPAR

GE 103: Introduction to Computing and Data Structures

First Semester of Academic Year 2024-2025

End Semester Examination

Dinesh 6/5/24

Duration: 3 Hours [2:30 PM – 5:30 PM] Max Points: 50 Date: 6-MAY-2024

Instructions:

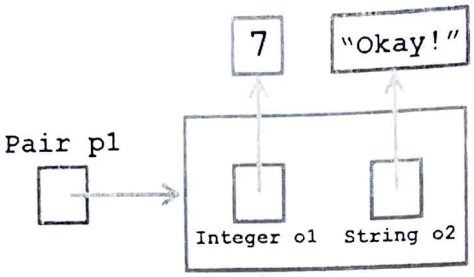
- There are 14 questions in the exam. All of the questions are mandatory.
- The points for each question are mentioned next to the question.
- Answer each question in the space provided for that question only.
- Structure your answer such that it does not go beyond the space allocated.
- No clarifications will be entertained during the examination. If you feel that a question is not clear, state your assumptions while answering.

Maximum Points	1.	2	3.	4.	5.	6.	7	8.
	3	3	4	4	5	2	4	4
Obtained Points	1	1+0	0	3	3	2	0	2
Maximum Points	9.	10.	11.	12.	13.	14.	Total	
	3	3	4	4	3	4		
Obtained Points	2	0	4	0	3	3.5	24.5	

Please Turn Over.

Q1. The below diagram shows an object p1 of the Pair class containing two values, one for each member variable associated with the object. Write the Object Oriented Python code corresponding to the diagram shown below. The following may be taken into account: [3 points]

A. You need to declare a class called Pair, with two member variables, as shown below.
B. Include a constructor function.




```
class Pair:
    def __int__ (self, integer O1, string O2):
        self.integerO1 = integerO1
```

```
class Pair:
    def __init__ (self):
        self.integerO1 = 7.
        self.stringO2 = "Okay!"
```

(1)

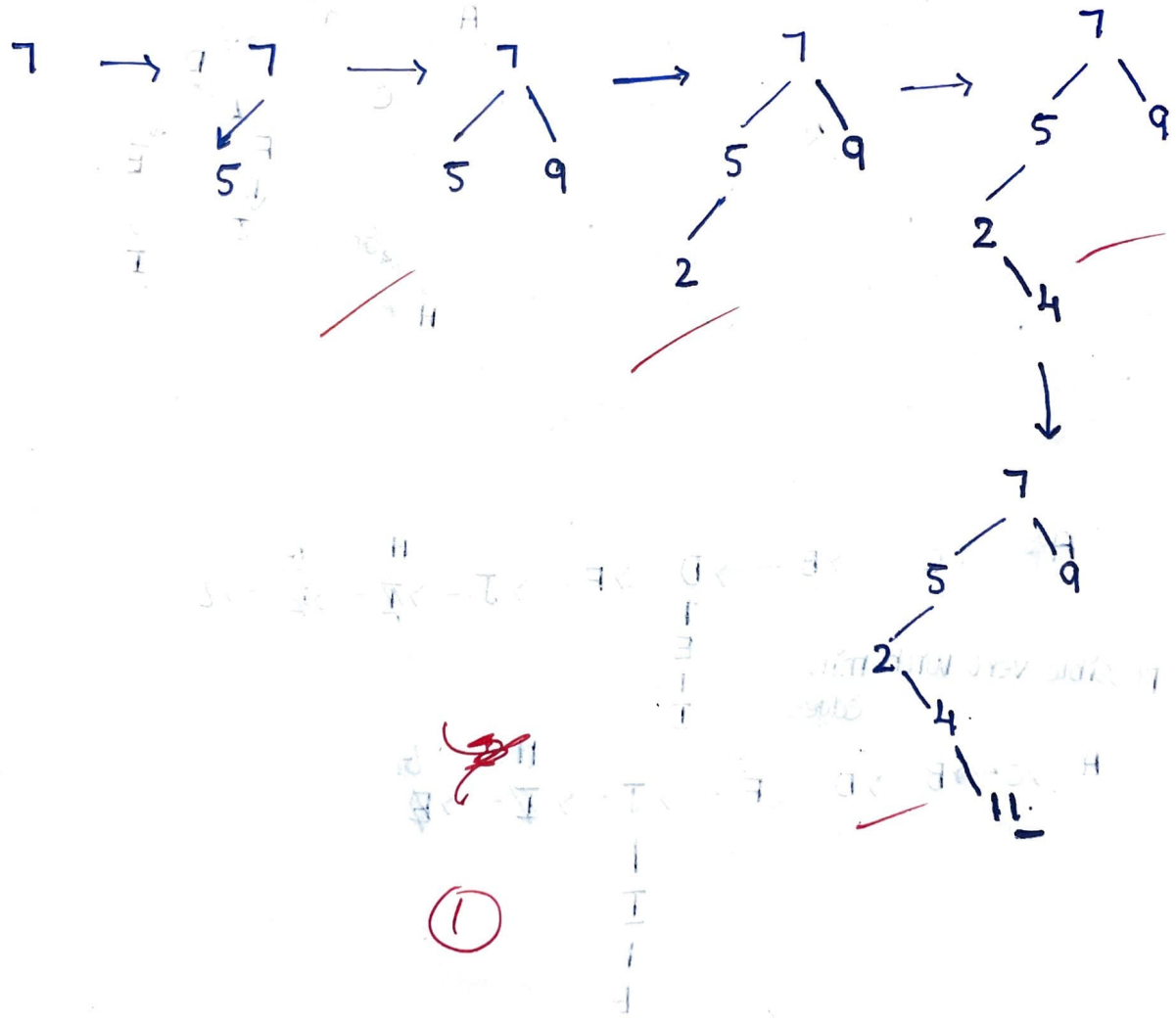
```
p=pair( )
print(p)
```

Q2. Binary Search Trees

A. Draw the resulting Binary Search Tree at **each** step after inserting and deleting the nodes in the following order – (i) insert 7. (ii) insert 5. (iii) insert 9. (iv) insert 2. (v) insert 4. (vi) insert 11. (vii) delete 4. Assume an empty tree at the beginning. Show the tree after performing each step. [2 points]

Binary search tree → Left node is always less than parent & right node always greater than parent

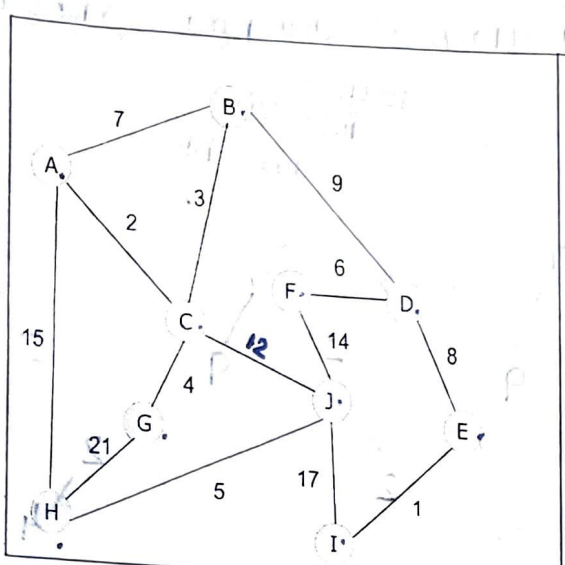
following
insert left
insert right



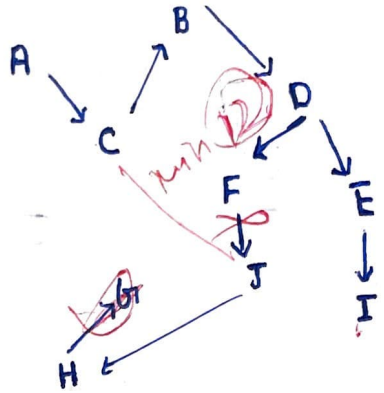
B. Is the final tree balanced? Answer with just 'yes' or 'no'. [1 point]

NO!

Q3. Draw the minimum spanning tree that is generated after applying Prim's algorithm to the graph below. Start with node A. Only draw the final minimum spanning tree. no explanation is required. [4 points]

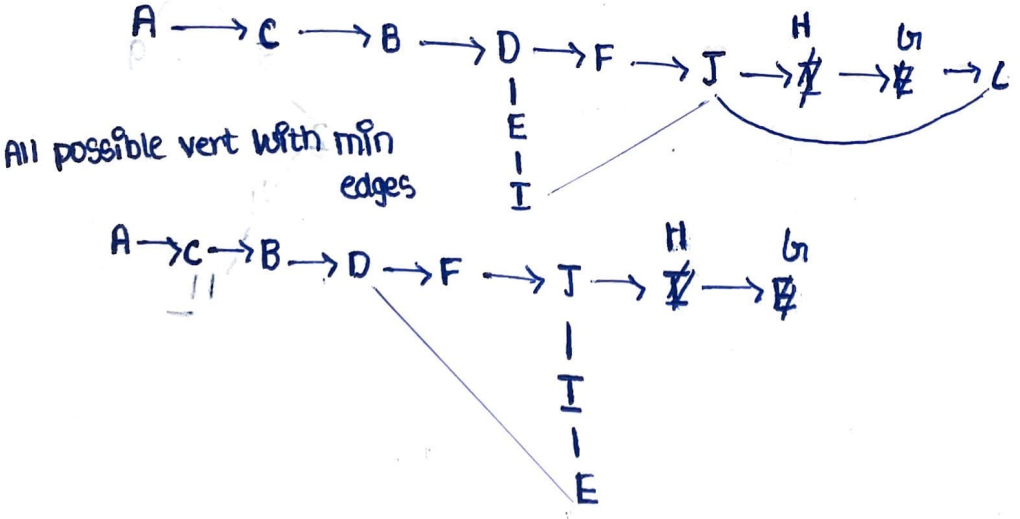


Final Answer:



10

Space for Rough Work:



Q4. Write down the expression of the worst case Big O complexity for the following operations. No explanation is necessary. [1 x 4 = 4 points]

A. `Contains` function: returns true or false depending on whether a particular object is a member of a list.

a. Sorted list implemented as an array. **Answer:** O(n)

b. Sorted list implemented as a single linked list (head reference only). **Answer:** O(n)

B. `Insert` function: add an object to the list in the appropriate place.

a. Unsorted list implemented as an array. **Answer:** O(n)

b. Sorted list implemented as an array. **Answer:** O(n)

Q5. **Circle the correct answer**. Only one answer correct per question. [1 x 5 = 5 points]

a) Which of the following functions grows the fastest? (b)

a. nlogn	(b) 2^n
c. logn	d. n^2

b) For a linked list implementation of a queue, if both front and rear have identical valid node values, the size of the queue is: (d)

a. 0	b. 1
c. 2	(d) the answer cannot be determined

c) For the linked implementation of a stack, where are the push and pop operations performed? (a)

(a) Push in front of first element, pop the first element	b. Push after last element, pop the last element
c. Push after last element, pop the first element	d. Push after first element, pop the first element

d) Which of the following does the binary heap implement? (b)

a. Binary search tree	(b) Priority queue
c. Stack	d. None of the above

e) Which of the following would require the most extra space, on average? (a)

(a) Bubble sort	b. Merge sort
c. Quick sort	d. Selection sort

Q6. List two applications each for Stacks and Queues. [2 points]

Stack (Application 1) Stacks are used in implementing depth first search algori	Stack (Application 2) Stacks are used for application involving backtracking & recursion in algorithms
Queue (Application 1) Queues are used in implementing breadth first Search algorithm in graphs	Queue (Application 2) Queues are used in computer networks, router, switch queues and email queues

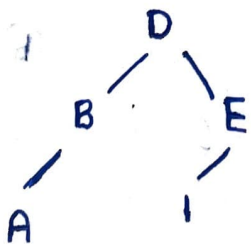
2

Q7. Consider the results of the following binary tree traversals: (i) Inorder = DBHEIAFCG, and (ii) Preorder = ABDEHICFG. LNR NLR

Construct a single corresponding binary tree that satisfies both these traversal sequences. [4 points]

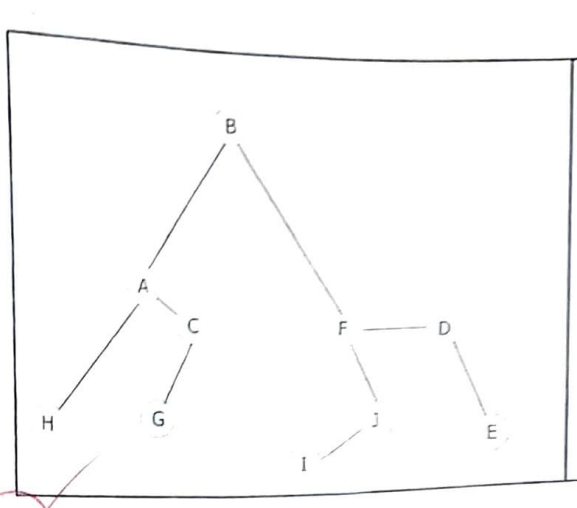
Inorder → root, left, right

preorder → left, root, right

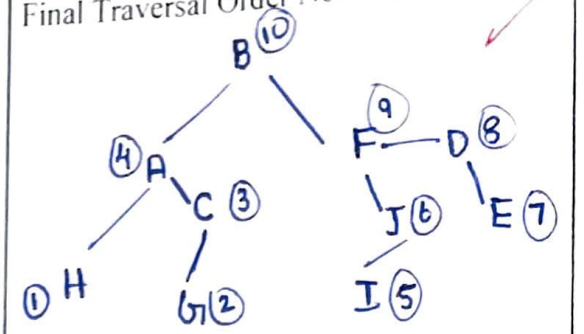


10

Q8. Consider the following graph. Draw the postorder traversal of the following binary tree, starting at node B. Show all the steps, and the contents of the stack at each step. [4 points]



Final Traversal Order Node by Node:



Stepwise Answer:

post order traversal → visits left then right then visits the root node

root node B → left → A → left → H (visited first)

↓ right

C → left → G (visited second)

↓ back tracking your parent

C (visited third)

↓ back tracking

A (visited fourth)

↓ right

F

left → J

↓ left

I (visited 5th)

↓ back tracking

J (6th)

right → D

↓ right

E (7th)

↓ backtracking

D (8th)

↓ backtracking

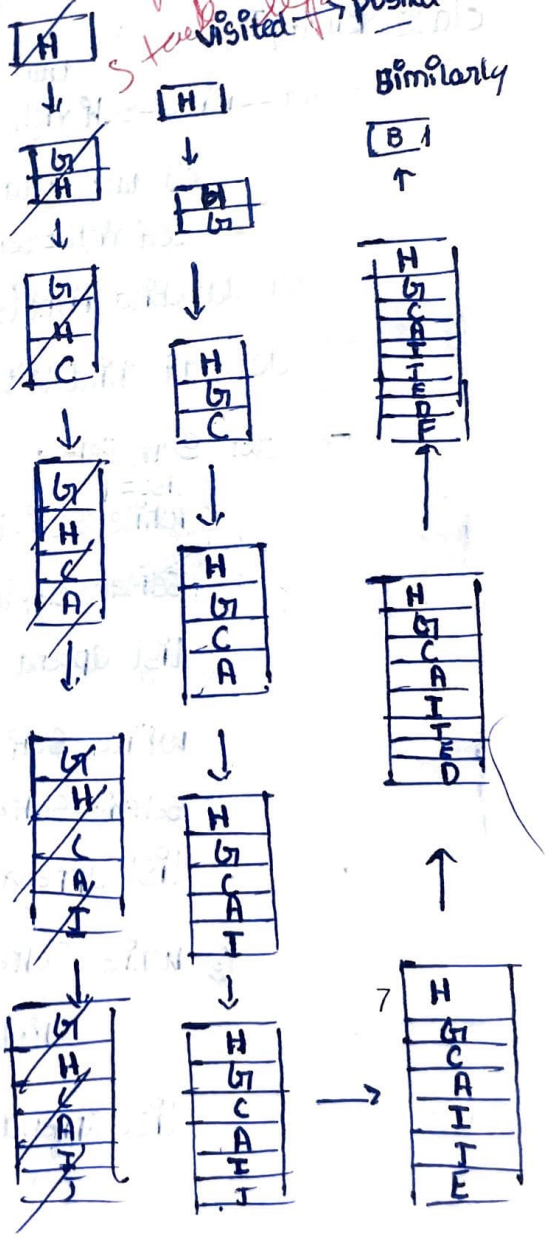
F (9th)

↓ To root node

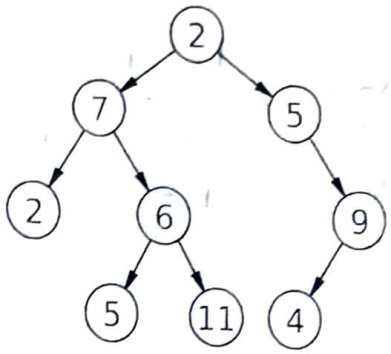
B (10th)

Stacks from B

visited → pushed



Q9. Given the definition of binary tree discussed in class. write the pseudocode for a **recursive** function named `SumLeaf( )`, that takes the root of a binary tree. and returns the sum of all the leaf nodes. For example. if you pass the root of the following binary tree. the function should return 2+ 5 + 11 + 4 = 22. [3 points]



Binary tree: A tree data structure with all the root/parent nodes having atmost two children and are termed as right and left child.

```
class BinaryTree:
    def __init__(self, data, right, left)
        self.data = data
        self.right = self.left = None
    def addchild right(self, child)
    def add child left(self, child)
    def Sum(self):
        list = [ ]
        ① while self.left:
            self.left = self.left.left
            list.append(self.left) → (2)
        ② while self.right:
            self.right = self.right.right
            list.append(self.right) → (4)
        ③ while self.left:
            self.left = self.left.right
            list.append(self.left) → (11)
```

(2)

list = [2,4,11,5]
for i in list
    sum += i

④ while self.left:
    self.left = self.left.right.left
    list.append(self.left) → (5)

Q10. Postfix notation

A. Write the following arithmetic expression in postfix notation: 5 + 3 * (2 + 4) Use the BODMAS (Brackets. Order. Division. Multiplication. Addition. Subtraction) rule for determining the order of operations. [1 point]

5+3*(2+4) → Prefix +5*3(+24) (or) maybe +5*3(.)+24

Ans +5*3+(24)

include brackets application in stack after addition of reversing tokens

B. Next, use a stack to evaluate the above postfix expression. Show the push and pop operations used for each input character step by step, and the resulting stack after each step. [2 points]

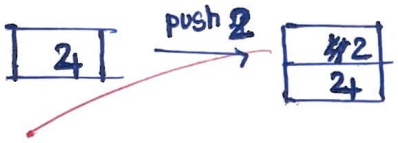
define class Stack:

with push, pop, is empty operations

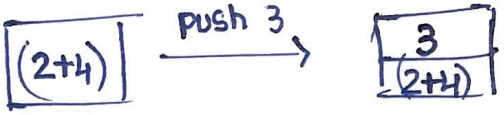
And proceed with expression evaluation

reverse tokens and push in if it is operand pop two out apply operator & push in if it is operator

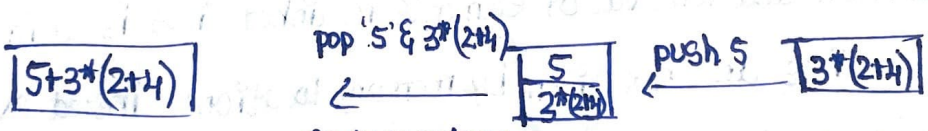
(2 4)+3*5+ → reversed



pop '2' and '4' apply operator + and ( ) and push in



pop '3' an (2+4) apply '*' a push



Apply operator + and push

↓ pop

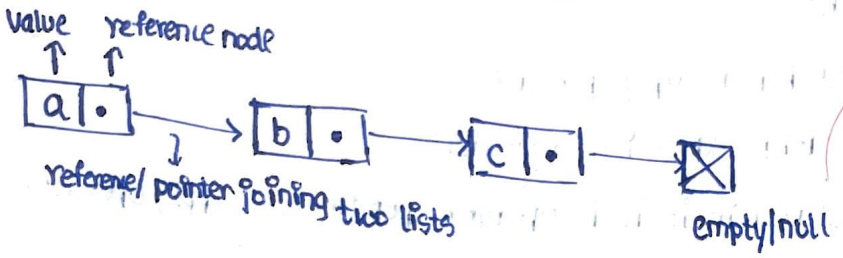
This is our required value

23

Q11. Linked Lists:

(a) What is a linked list data structure? Explain with the help of a pictorial representation. [2 points]

1. Linked lists are the data structure which contain nodes in which one of the node contains value and other contains reference to the next node
2. In linked lists unlike arrays each element points to the next
3. Due to this property of linked lists addition and removal of elements is easier
4. Search time in linked lists is linear



2

(b) Give one advantage and one disadvantage of the linked list data structure over an array. [2 points]

Advantage:

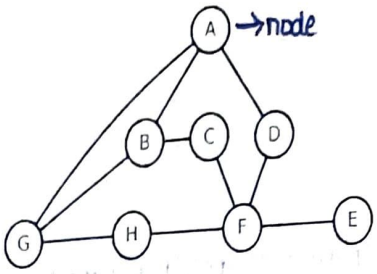
1. Addition and removal of elements in linked lists is easier than of array since linked lists are not given by memory locations insted every element points to the next.
2. Linked lists are used to implement queues and stacks more efficiently than lists

Disadvantage:

1. The search time over linked lists is linear. O(n)
2. Arrays are more abstract classes than linked lists which make the life of user easy.

2

Q12. Given the following **graph**:



Explain and show the steps for breadth-first search traversal using the ~~stack~~ queue data structure, starting from node A.

[4 points]

Breadth first search:

```
class Treenode:
    def __init__(self,data)
        self.data= data
        self.left= None
        self.right =None
```

Implement class Queue():

```
class BFS
def bfs_traversal(Node)
    if Node= None
        return
    Queue= Queue()
    queue.enqueue(node)
    visited = [ ]
```

next

```
while not queue.isempty()
    current_node = queue.dequeue()
    visited.append(current node)
if queue.is left:
    queue.enqueue(queue.is left)
if queue.is right
    queue.enqueue(queue.is right)

return visited
```

The queue initially contains [A] ⇒ visited = [A]

left of root A = B → enqueued & visited

right of A = C → visited at same depth

right of A = D → visits

left of B = G → visits

at same depth of B = H → visits

at left of D = F

at same level of F = E

visited = [A,B]

visited = [A,B,C]

visited = [A,B,C,D]

visited = [A,B,C,D,G]

[A,B,C,D,G,H]

[A,B,C,D,G,H,F]

[A,B,C,D,G,H,F,E].

This is DFS

completes

Q13. Mutability:

(a) Give an example to illustrate the difference between mutable and immutable data types in Python. [2 points]

Mutable data types: The data types which can be changed (or) modified without changing memory address Ex: lists, dictionaries, sets

Immutable data types: Memory adress changes on changing data types Ex: tuple, integers, float, bool

Ex:

(Mutable)

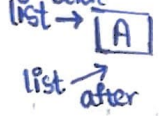
List = [1,2,3]

hex(id(list)) = A

list.append(4)

>> list = [1,2,3,4]

hex(id(list)) = A



(Immutable)

tuple = (1,2,3)

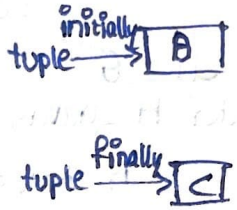
b = (4,)

hex(id(tuple)) = B

tuple = tuple + b

>> tuple = (1,2,3,4)

hex(id(tuple)) = C



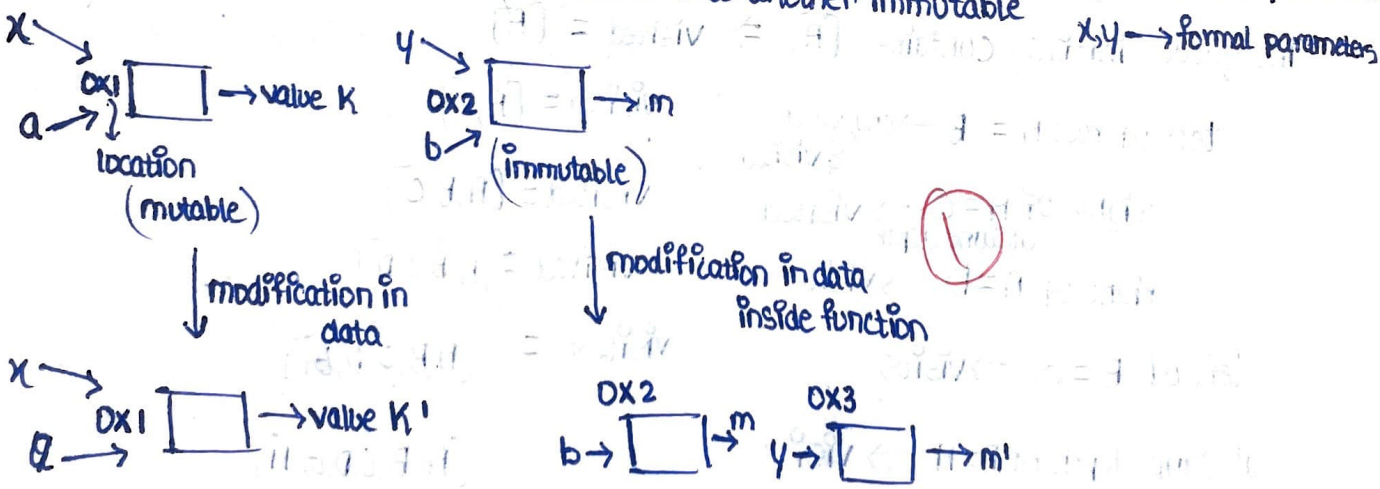
(b) How does the mutability of function arguments affect the behavior of functions in Python? [1 point]

In functions we know when the function is called the formal parameters are bound to the actual parameters.

Consider '2' parameters one mutable and another immutable

a, b → actual parameters

x, y → formal parameters



For mutable arguments any change in arguments inside function also affects the actual parameter outside function. For immutable arguments both formal & actual parameters are distinct.

Q14. Access Modifiers:

A. List and explain the different access modifiers available in Python with an example. [2 points] Different access_modifiers available in python are private & protected (public?)

- private members can be only accessed within the class cannot be accessed in the child class (or) outside (& modified) (driven by adding double underscore before variable.
- protected members are only accessed and changed within parent and child classes (single underscore)

Ex:

```
class Parent:
def __init__(self):
    self.__d = 42

class Child(Parent):
    def __init__(self):
        Parent.__init__(self)
    def display(self):
        print(self.__d)
```

#error

```
O2 = parent()
O1 = child()
O1.display()
O2.self.__d)
```

1 & 2 produce error

```
class Parent:
def __init__(self):
    self._d = 42
class Child(Parent)
    def __init__(self):
        Parent.__init__(self)
    def display(self)
        print(self._d)  # will work
```

1.5

B. What is inheritance property in the Python OOP paradigm? List two advantages of the inheritance property. [2 points]

Inheritence is the property (or) capability of a class to derive (or) inherit the properties of other class.

The class which inherits the properties is called child/subclass and from which properties are inherited is the parent/super class.

The child class derives all the properties from parent class and can also have methods defined unique to itself.

Advantages:

2

1. Inheritence provides reusability of code.
2. Inheritence represents the realworld relationships well.
3. Inheritence helps to reduce the amount of code to be written in child class providing a level of abstraction.