INDIAN INSTITUTE OF TECHNOLOGY ROPAR

GE 103: Introduction to Computing and Data Structures

First Semester of Academic Year 2024-2025

End Semester Examination

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

Instructions:

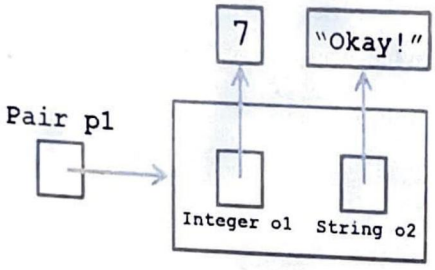
- There are 14 questions in the exam. All of the questions are mandatory.
- The points for each question are mentioned next to the question.
- Answer each question in the space provided for that question only.
- Structure your answer such that it does not go beyond the space allocated.
- No clarifications 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								
Maximum Points	9	10	11	12	13	14	Total	
	3	3	4	4	3	4		
Obtained Points								

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.



Ans:

```
Class Pair1:
    def __init__(self, Integer, String):
        self.integer = Integer
        self.string = String
    def show(self):
        print("Integer is", self.integer)
        print("String is", self.string)

p1 = Pair(7, "Okay!")
p1.show()
```

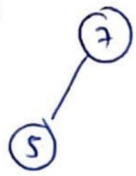
Q2. Binary Search Trees

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]

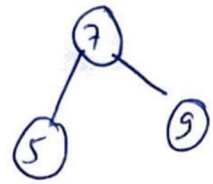
Ans:-

(i)

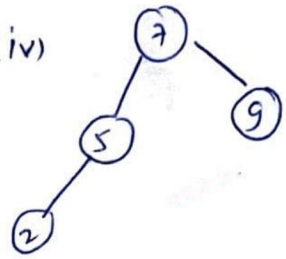
(ii)



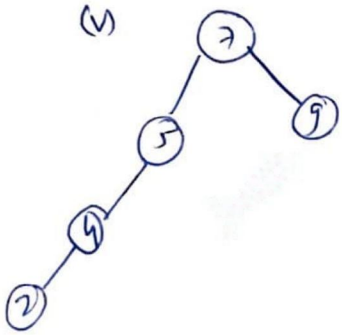
(iii)



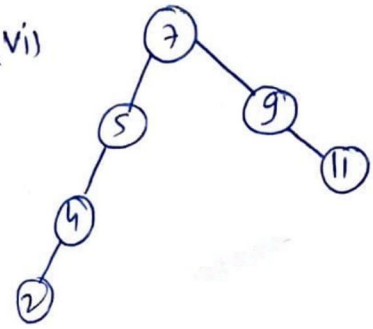
(iv)



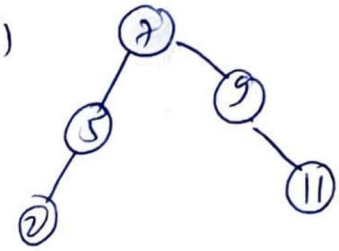
(v)



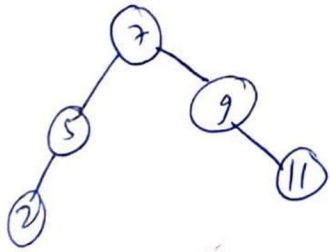
(vi)



(vii)



final tree :

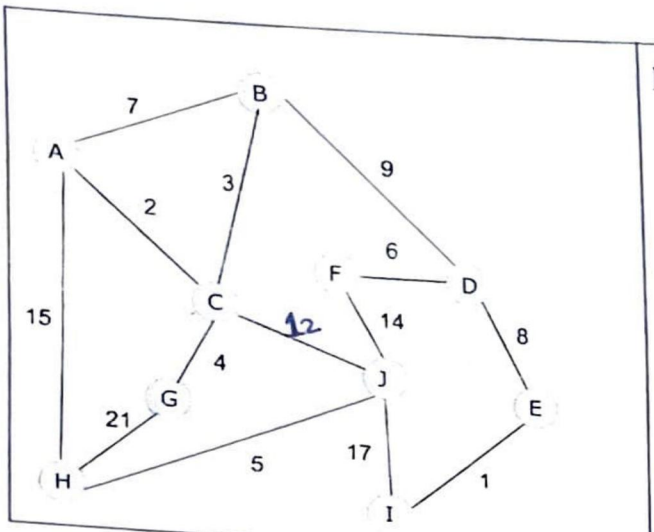


B. Is the final tree balanced? Answer with just 'yes' or 'no'. [1 point]

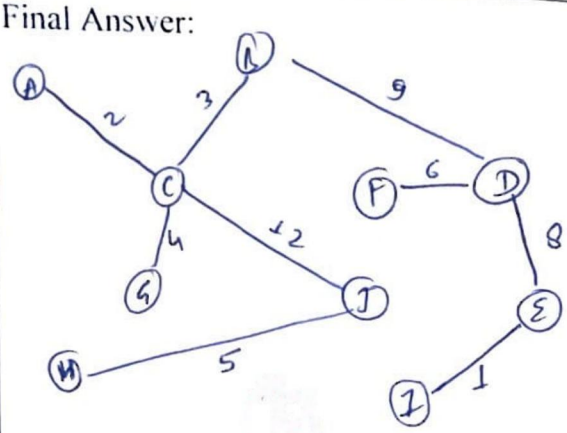
→ Yes.

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.

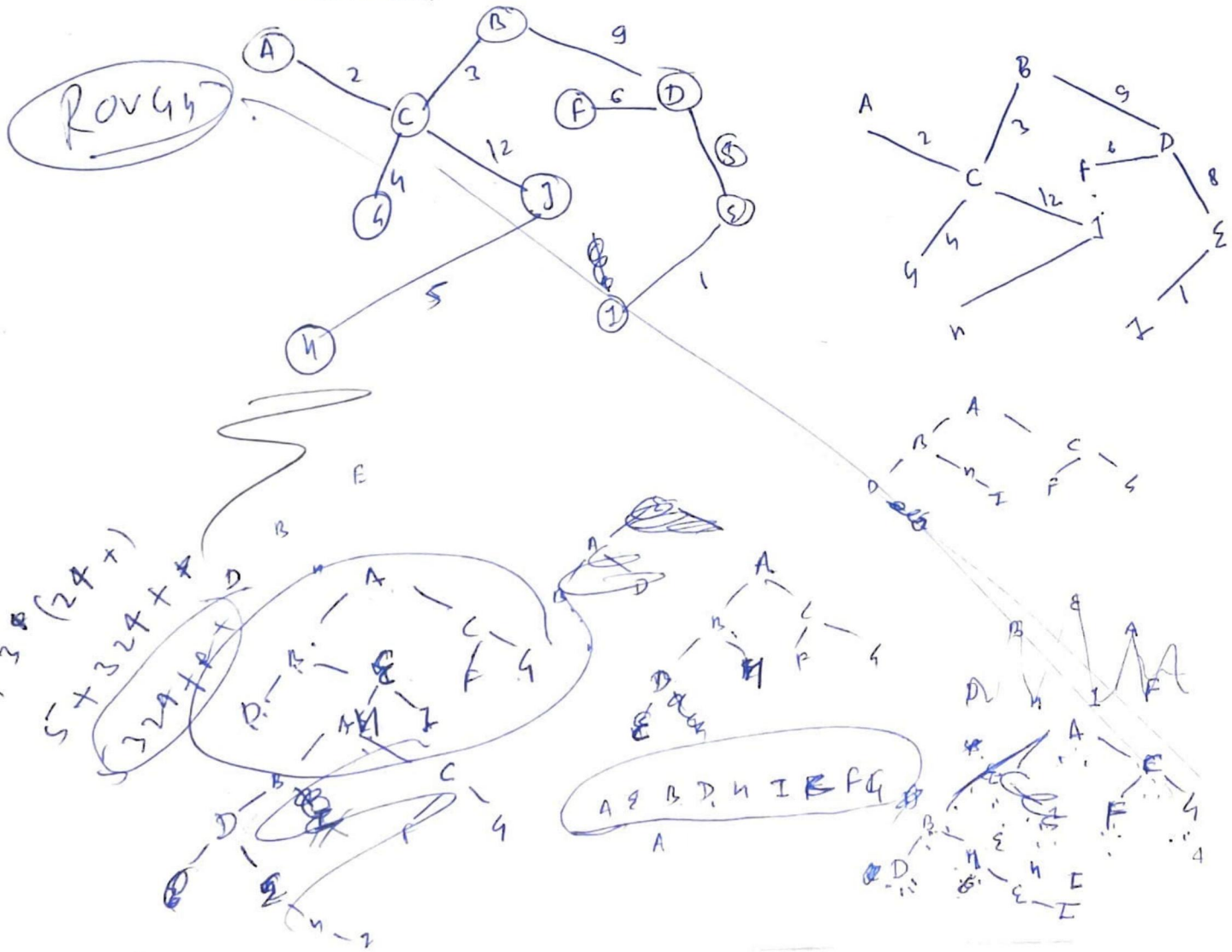
[4 points]



Final Answer:



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(1)

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(1)

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?

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:

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. 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?

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. Bubble sort	b. Merge sort
c. Quick sort	d. Selection sort

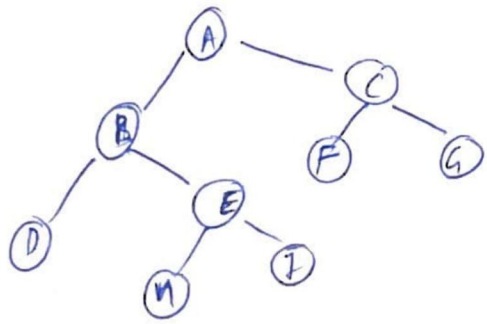
Q6. List two applications each for Stacks and Queues. [2 points]

Stack (Application 1)	Stack (Application 2)
• Stack is used in Depth first Search (DFS)	• Stack is also used in postfix to infix Conversion.
Queue (Application 1)	**Queue (Application 2)**
• Queue is used in Breadth first search (BFS)	• Queue is used in "matching ~~comparing~~ parenthesis".

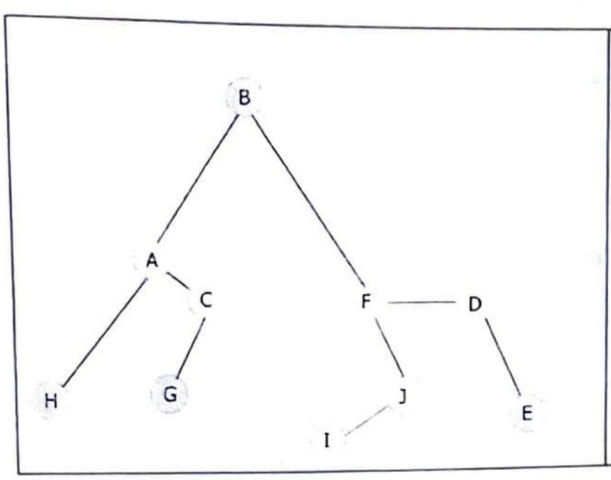
Q7. Consider the results of the following binary tree traversals: (i) Inorder = DBHEIAFCG, and (ii) Preorder = ABDEHICFG.

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

ans;



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:

H G C A I J G D F B

Stepwise Answer:

(i) ans: Stack

Stack ← S: | B |

List of post order traversal ← L:

(ii) S: | B | A |

L:

(iii) S: | B | A | H |

L:

(iv) S: | B | A |

L: | H |

(v) S: | B | A | C |

L: | H |

(vi) S: | B | A | C | G |

L: | H |

(vii) S: | B | A | C |

L: | H | G |

(viii) S: | B | A |

L: | H | G | C |

(ix) S: | B |

L: | H | G | C | A |

(x) S: | B | F |

L: | H | G | C | A |

(xi) S: | B | F | J |

L: | H | G | C | A |

(xii) S: | B | F | J | I |

L: | H | G | C | A |

(xiii) S: | B | F |

L: | H | G | C | A | I | J |

(xiv) S: | B | F | D |

L: | H | G | C | A | I | J |

(xv) S: | B | F | D | E |

L: | H | G | C | A | I | J |

(xvi) S: | B | F |

L: | H | G | C | A | I | J | G | D |

(xvii) S: | B |

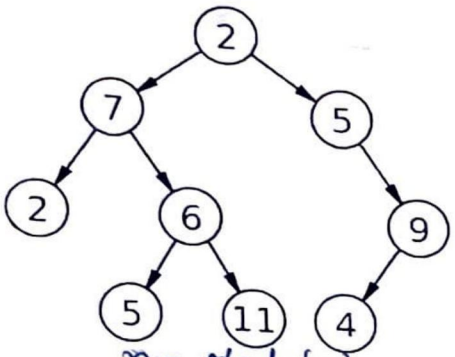
L: | H | G | C | A | I | J | E | D | F |

(xviii) S:

L: | H | G | C | A | I | J | E | D | F | B |

Post order traversal

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]



~~S = 0~~

ans:-

```
n = Node(2)
S = 0
def SumLeaf(n):                    # left child of node n ; right child of node n
    if self.left_child is None and self.right_child is None:
        ~~return~~
        S += self.data
        return S
    ~~return SumLeaf(self.leftchild)~~
    if n = None:
        return 0
    return SumLeaf(self.left_child) + SumLeaf(self.right_child)

~~Print(~~
~~SumLeaf(n))~~
Print(SumLeaf(n))
```

Q10. Postfix notation

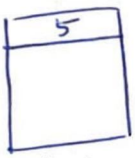
A. Write the following arithmetic expression in postfix notation: 5 + 3 * (2 + 4) Use the BODMAS (**B**rackets, **O**rder, **D**ivision, **M**ultiplication, **A**ddition, **S**ubtraction) rule for determining the order of operations. [1 point]

ans:- 5324+*+

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]

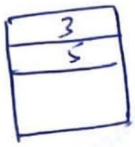
ans:-

(i) input = 5

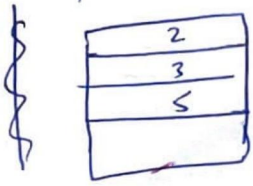


Stack

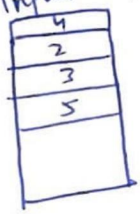
(ii) input = 3



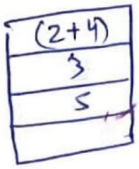
(iii) input = 2



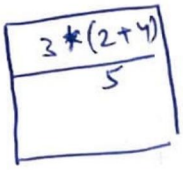
(iv) input = 4



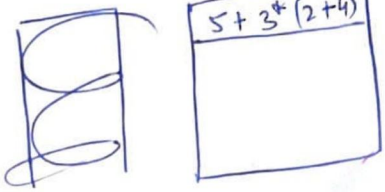
(v) input = +



(vi) input = *



(vii) input = +

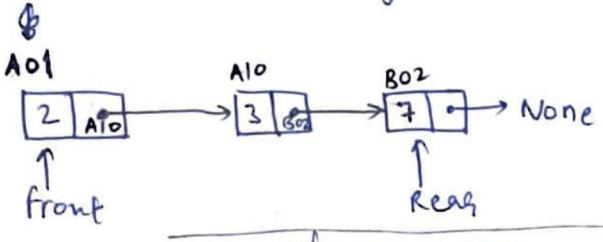


So, This is the resulting stack after each step

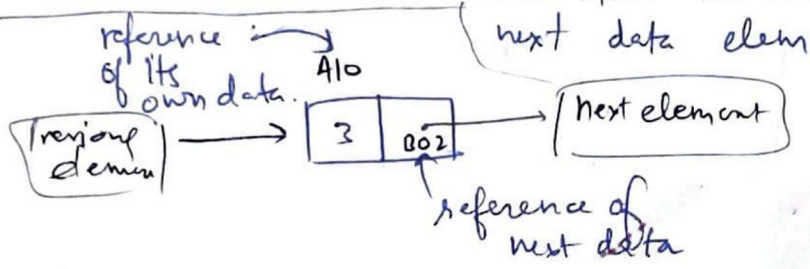
Q11. Linked Lists:

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

ans:- The linked list data structure is the list of elements which stores the data value as well as the reference of the adjacent / next data value.



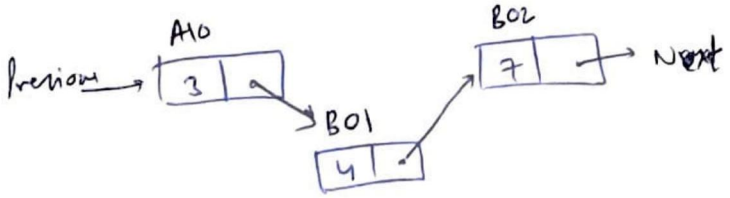
The adjacent list shown can be seen that each element have value their value and stores the address of the next data element



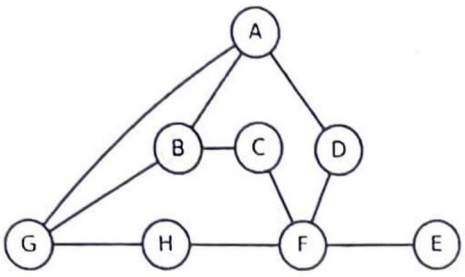
(b) Give one advantage and one disadvantage of the linked list data structure over an array. [2 points]

ans:- Disadvantage :- When you have to add the element at the end of list, then the iteration will go over the list till the end for finding last element, and then add the new element after that element.

Advantage :- It has advantage over an array in insertion of an element. It will directly change the reference and then element is added.



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]

Ans:- (i) Qnew

Qnew → Q: | A |

visited

visited → V: | A |

(ii)

Q: | G | B | D |

V: | A | G |

(iii)

Q: | B | D | H |

V: | A | G | B |

(iv)

Q: | D | H | C |

V: | A | G | B | D |

(v)

Q: | H | C | F |

V: | A | G | B | D | H |

(vi)

Q: | C | F |

V: | A | G | B | D | H | C |

(vii)

Q: | F |

V: | A | G | B | D | H | C | F |

(viii)

Q: | E |

V: | A | G | B | D | H | C | F | E |

BFS traversal :- A G B D H C F E

In BFS traversal, ~~the~~ first all the nodes at the same level is visited then moving onto the another level.

Q13. Mutability:

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

ans:-

```
l1 = [1, 2, 3]
l1.append(4)
print(l1)
```

Output:-

[1, 2, 3, 4]

→ l1 is the list.
→ You have changed the original list
→ list is mutable
→ Python will change the list l1 at the same memory address.

~~T = (1, 2, 3)~~

```
x = 3
print(id(x))
x += 4
print(id(x))
```

If you run this in ~~com~~ compiler, the both id(x) will be ~~diffi~~ different. Because int datatype is immutable and python ~~over~~ ~~the different~~ makes variable x to point to diffrent memory address containing 7.

→ int datatype is immutable

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

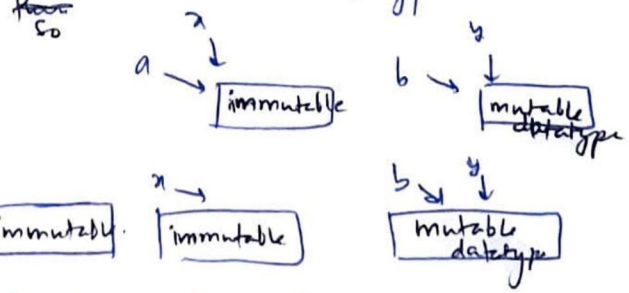
ans:-

```
def f(x, y):
    y.append(4)
    x += 4
    return None

f(a, b)
print(a)
print(b)
```

let say a is immutable int & b is mutable list datatypes.

~~Hoo~~ Co



then

In the output of this code snippet, the value of a remains unchanged but the list b will get change because it was mutable.

Q4. Access Modifiers:

A. List and explain the different access modifiers available in Python with an example. [2 points]

ans:

- Private

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

here d is private instance variable.

- You cannot access the value of d with the help of subclass of Home or ~~outside~~ directly outside.
- Use of double underscore

- Protected

```
class Home():
    def __init__(self):
        self._d = 42
```

here, d is protected instance variable

- You can access the value of d with the parent class and with the child class but not directly outside.
- Use of single underscore

- Public

```
class Home():
    def __init__(self):
        self.d = 42
```

here, d is public instance variable.

- You can access the value of d from anywhere.

~~Use of~~

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

ans: The inheritance property in Python OOPs allow you to inherit the property of ~~the~~ parent class into the subclass ~~class~~ called as child class.

Advantages:-

- It allows you to ~~reduce~~ use the variable and methods which are defined in parent class, without defining again in the child class.
- It reduces the amount of code ~~and makes the code more~~ from repetition.

```
class Parent():
    def __init__(self, age):
        self.age = 37

class Child(Parent):
    def __init__(self):
        Parent.__init__(self)
        self.name = "Arjun"
```