Complete the recursive function **sortedMerge** which will take head of two sorted linked lists along with head of the new list. The function should create a new sorted linked list in **ascending** order and return the head of the new list. Assume that the lists will not be empty. **You do not need to create any new Node.**

Python notation:

```
def sortedMerge (h1, h2, newHead):
        #To do
```

Java notation:

```
public Node sortedMerge (Node head1, Node head2, Node newHead) {
        // To do
}
```

Sample Input	Sample Output
List 1: 10 → 25 → 30 → 40 → None List 2: 5 → 7 → 20 → 27 → None	5 → 7 → 10 → 20 → 25 → 27 → 30 → 40 → None

---

```
def sortedMerge(h1, h2, newhead):
    if h1 is None:
        newhead.next = h2
        return newhead
    elif h1.element > h2.element:
        newhead.next = h2
        return sortedMerge(h2, h2.next, newhead.next)
    elif h2.element > h1.element:
        newhead.next = h1
        return sortedMerge(h1.next, h2, newhead.next)
    elif h2 is None:
        newhead.next = h1
        return newhead
```

Draw the contents of the hash table given the following conditions.

- The size of hash table is 8
- Linear Probing is used to resolve collisions
- H(K) is the hash function where K is length of given string.

IF K is even then
    R(K) = K * 2
Else
    R(K) = K * 3
IF R(K) > 10 then
    H(K) = R(K) % 5
Else
    H(K) = R(K) % 2

Insert the following values in the hash table. Show all the collision.

HELLO DARKNESS MY OLD FRIEND

Start writing your answer below the line.

10

Sln ⇒ Size of hash table = 8
∴ Index of array is 0 to 7

String	K	R(K)	H(K)	Actual index
HELLO	5	15	0	0
DARKNESS	8	16	1	1
MY	2	4	0	2 [collision]
OLD	3	9	1	3 [collision]
FRIEND	6	12	2	4 [collision]

Arr = 0 Hello, 1 DARKNESS, 2 MY, 3 OLD, 4 FRIEND, 5, 6, 7

(Ans).

[No extra sheet will be provided. Write your answer to the questions in this answer script.]
[Marks allocated to each question is given in the statement of corresponding question.]

1. **Consider the following recurrence relation with the base condition $a_0=2$, $a_1=1$, $a_2=3$**

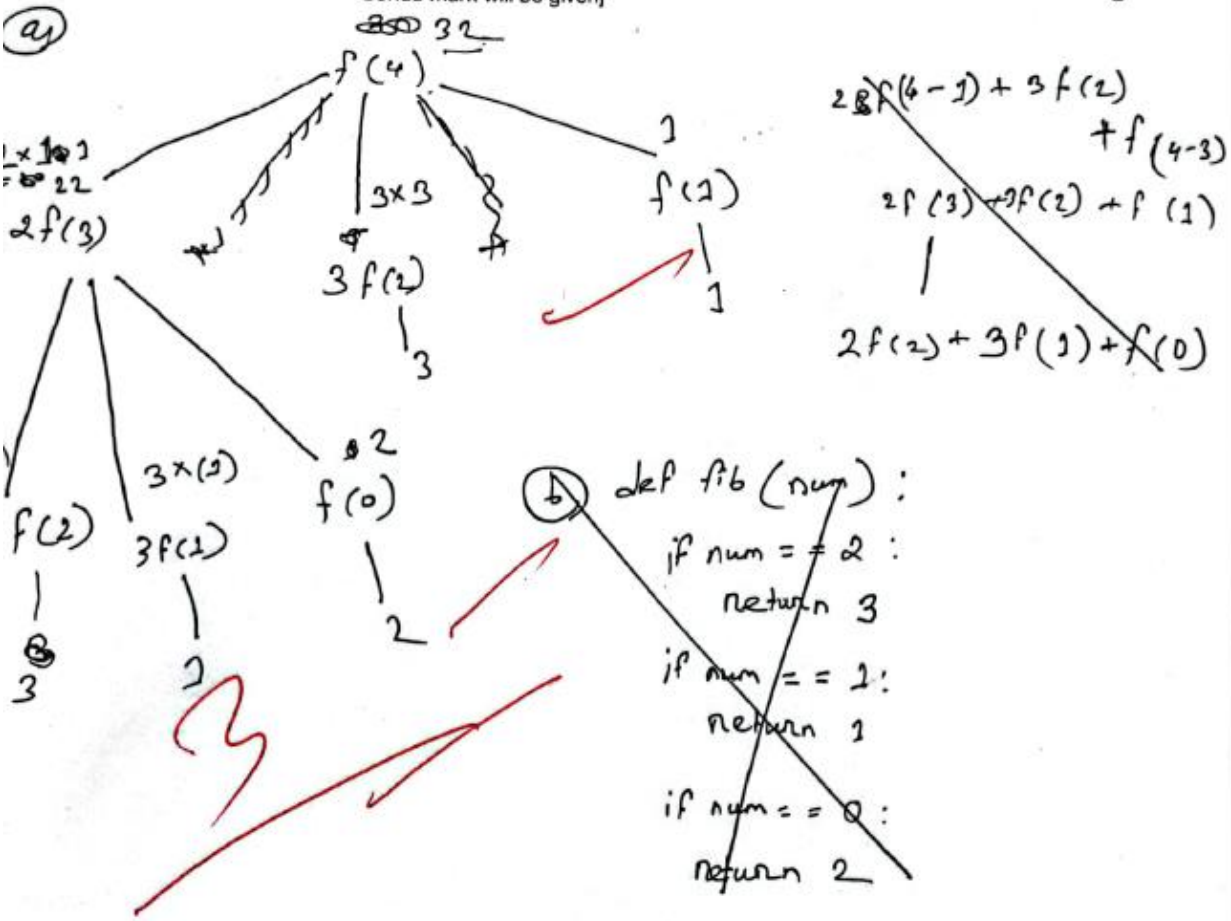
$$f(n) = 2 * f(n-1) + 3* f(n-2) + f(n-3)$$

a. Draw a recursion tree of f(4) for the above recurrence relation and evaluate it at its root. 3

b. Write a recursive function that finds out the n-th number of the sequence. 3

c. Write an iterative function with memorisation to combat overlapping subproblems for the same recurrence relation. 4

[If you can solve the iterative approach with three variable instead of an array, bonus mark will be given] 2



```
def fib (num):
    if num == 2:
        return 3
    if num == 1:
        return 1
    if num == 0:
        return 2
    else:
        return 2*fib(num-1) + 3*fib(num-2)
                            + fib(num-3)

temp = fib(4)
print (temp)
```

```
def fib (num):
    array1 = [None] * num
    for i in range (0, num):
        if i == 0:
            array1[i] = 2
        if i == 1:
            array1[i] = 1
        if i == 2:
            array1[i] = 3
        else:
            array1[i] = 2*i
                        + 3*(i-2)
                        + i-3
temp = fib (num)
print (temp
```

```
def fib (num):
    array1 = [None] * num
    sum = 0
    for i in range (0, num):
        if i == 0:
            array1[i] = 2
        if i == 1:
            array1[i] =
temp = fib (4)
print (temp)
```

for i in range (4)

$a_0 = 2$

$a_1 = 1$

$a_3 = a_0 + a_1$

[No extra sheet will be provided. Write your answer to the questions in this answer script.]
[Marks allocated to each question is given in the statement of corresponding question.]

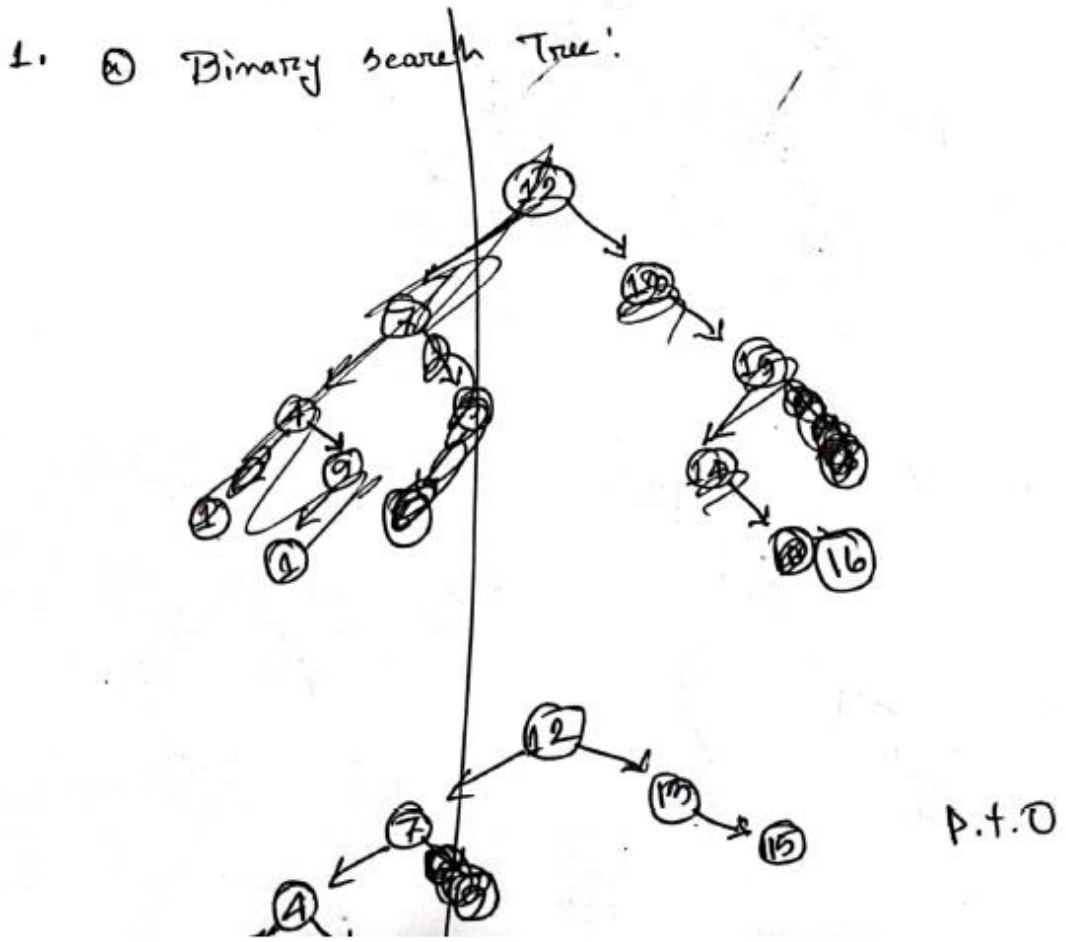
1. Consider the following ten numbers:

12, 7, 4, 13, 15, 9, 1, 14, 8, 16

a. Draw Binary Search Tree by inserting the above numbers as keys of the nodes from left to right. 4

b. Write down the keys of the above binary search tree in **preorder, inorder and postorder** traversal. 3

c. Delete the node of the key containing **7** from the above binary search tree. 3
[You must show each process step by step]



1.

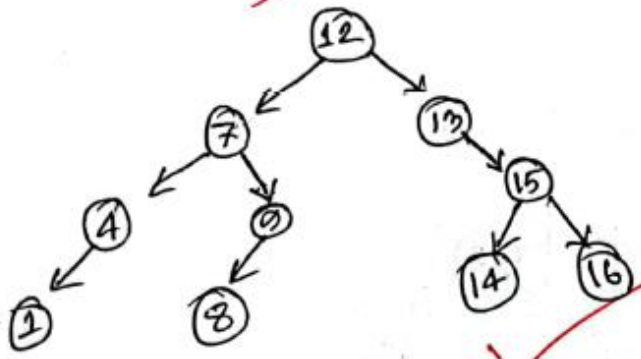
(b)

pre-order : ~~12, 7, 4, 1,~~ 9, 8, 13, 15, 14, 16

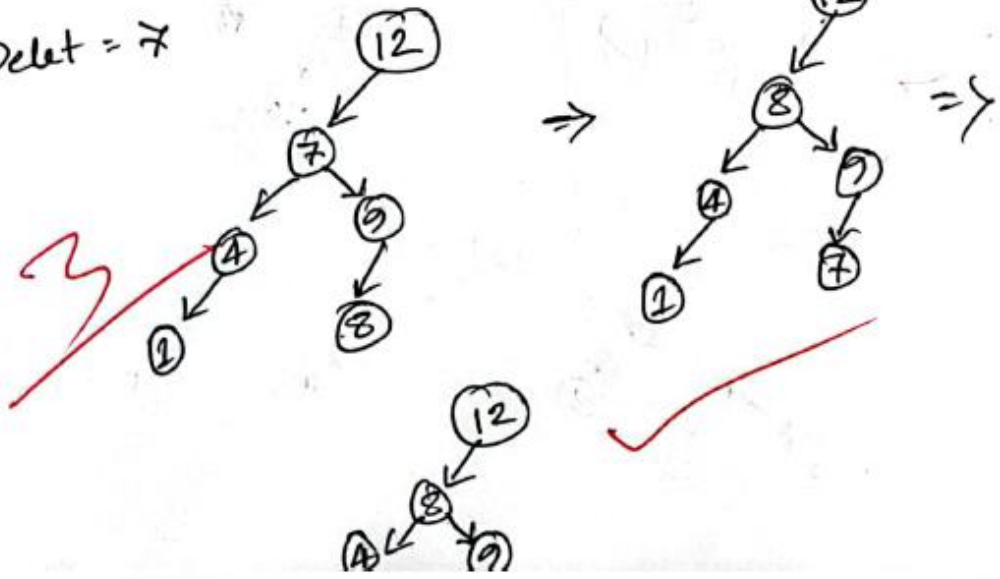
In-order : 1, 4, 7, 8, 9, 12, 13, 14, 15, 16

Post order : 1, 4, 8, 9, 7, 14, 16, 15, 13, 12.

1. (a)



(c) Delet = 7



1. Consider the following adjacency matrix for an undirected graph.

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	1
C	0	1	0	0	1
D	1	0	0	1	0
E	1	1	0	0	0

a. What is the course code of this course? – 2
b. Is the adjacency matrix given above correct? If not, modify it into a correct one. – 4
c. Find the adjacency list from the given adjacency matrix representation. – 4

Answer to the question no.01

(a)

Course code of this course : CSE220

Name of the course : Data Structures.

(b)

The adjacency matrix given above is not correct becaus
The diagonal of the matrix should have '0' as elements.

P.T.O

It should have symmetrical values for an undirected graph.

Modification 2

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	1
C	0	1	0	0	1
D	0	1	0	0	0
E	1	1	1	0	0

(c)

Adjacency list according to the modification in (b):

A → B, E

B → A, C, D, E

C → B, E

D → B,

E → A, B, C

Set of adjacency list: Vertices and Edges:

(V, E) = (A, B, C, D, E), {(A, B), (A, E), (B, C), (B, D), (B, E), (C, E), (A, C)}.

1. [CO1] Consider the following circular queue:

Q = [7, 6, none, none, 5, 10, 14]

a) *[2 marks]* Which element is the **front** and which element is the **rear/back** of Q?

Ans: front → 5
rear → 6

2

b) [6 marks] After executing the following operations, write down the correct output respectively:

- dequeue() **Ans:** [7, 6, none, none, none, 10, 14]
- dequeue() **Ans:** [7, 6, none, none, none, none, 14]
- peek() **Ans:** 14
- enqueue(9) **Ans:** [7, 6, 9, none, none, none, 14]
- enqueue(200) **Ans:** [7, 6, 9, 200, none, none, 14]
- dequeue() **Ans:** [7, 6, 9, 200, none, none, none]

6

2. [CO2] *[5 marks]* Write a recursive function for binary search algorithm.

def binSearch(arr, left, right, key):

//Write code here

(arr is a sorted array, left is the left index, right is the right index, key is the element which we are searching in the array)

3. [CO2] *[2 marks]* Consider the following sorted array:

arr= [5, 10, 15, 20, 30, 40, 50]

Draw the recursion tree of ***binSearch(arr, 0, 6, 15).***

binSearch (arr, 0, 6, 15)
↓ mid = 3
binSearch (arr, 0, 2, 15)
↓ mid = 1
binSearch (arr, 2, 2, 15)
↓ mid = 2
True

2

② def binSearch(arr, left, right, key):

```
    if (left > right):
        return False

    else:
        mid = (left + right)//2
        if (arr[mid] == key):
            return True
        elif (arr[mid] < key):
            left = mid + 1

        else: right = mid - 1

        return binSearch(arr, left, right, key)
```

1. [CO2] Given the array representation of a binary tree.
[null value means the node is empty]:
[null, 12, 15, 3, 25, 13, null, 23, null, -2, null, 0, null, null, -5, -9]

a) Draw the binary tree. **[2 marks]**

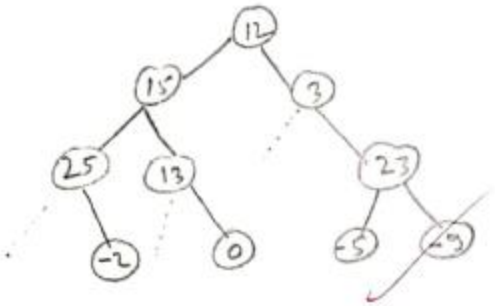
b) Write the pre-order, post-order and in-order traversal sequence of the tree. **[3 marks]**

c) Convert the tree to a complete binary tree. **[1 mark]**

d) Use the post order traversal sequence in part (b) to insert the elements in that order in an initially empty binary search tree, and show the resulting binary search tree.
Note: Consider the first element of the post order sequence as the root. **[2 marks]**

e) Perform the following operations step by step on the Binary Search Tree you created in part d). **[2 marks]**
i. Delete node -2 with the help of its successor.
ii Delete node 13 with the help of its predecessor.

a)



2

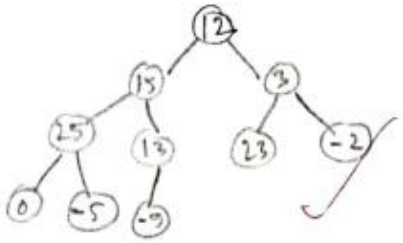
b) pre-order : (Root → left → Right) → 12 15 25 -2 13 0 3 23 -5 -9

post-order : (Left - Right → Root) → -2 25 0 13 15 -5 -9 23 3 12

In-order: (Left - Root - Right) → 25 -2 15 13 0 12 3 -5 23 -9

3

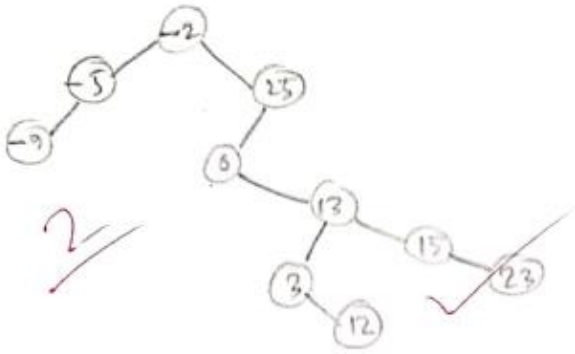
(c) complete binary tree:



1

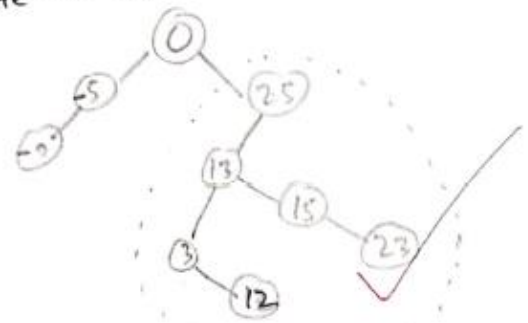
(d) post-order traversal from (b)

-2, 25, 0, 13, 15, -5, -9, 23, 3, 12



2

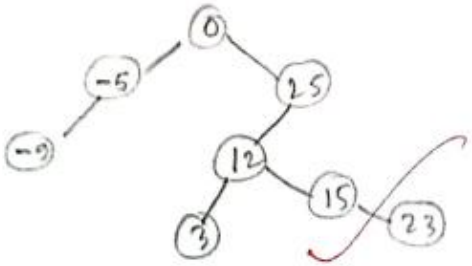
(e) (i) delete -2 by successor



2

successor
→ Right subtree
2' Min
→ Pred -
Left subtree go
largest

(ii) delete 13 by predecessor.



Complete the recursive function **sortedMerge** which will take head of two sorted linked lists along with head of the new list. The function should create a new sorted linked list in **descending** order and return the head of the new list. Assume that the lists will not be empty. **You do not need to create any new Node.**

Python notation:

```
def sortedMerge (h1, h2, newHead):
        # To do
```

Java notation:

```
public Node sortedMerge (Node head1, Node head2, Node newHead) {
        // To do
}
```

Sample Input	Sample Output
List 1: 40 → 30 → 25 → 10 → None List 2: 27 → 20 → 7 → 5 → None	40 → 30 → 27 → 25 → 20 → 10 → 7 → 5 → None

---

```
def sorted Merge (h1, h2, newHead):
    if h1 == None and h2 == None:
        newHead.next = None
        return
    elif h1 == None and h2 != None:
        newHead.next = h2
        sorted Merge (h1, h2.next, newHead.next)
    elif h2 == None and h1 != None:
        newHead.next = h1
        sorted Merge (h1.next, h2, newHead.next)
    else:
        if h1.elem > h2.elem:
            newHead.next = h1
            sorted Merge (h1.next, h2, newHead.next)
        else:
            newHead.next = h2
            sorted Merge (h1, h2.next, newHead.next)
    return newHead
```

1. Suppose, you are given a list *l* (You do not need to take any input). Write a recursive function that reverses the list *l*.

Sample Input: [1,2,3,4,5] Sample Output: [5,4,3,2,1]	Sample Input: ['c', 's' ,'e' ,2, 2, 0] Sample Output: [0, 2, 2, 'e', 's', 'c']

[6]

```
def reverse(arr, n=0):
    if n > int(len(arr)/2):
        return arr

    temp = arr[n]

    arr[n] = arr[len(arr)-1-n]
    arr[len(arr)-1-n] = temp

    reverse(arr, n+1)
```

2. Suppose, you have been given a non-negative integer n. Write a recursive function prints all the **Hailstone Numbers** from n upto 1. Hailstone number follows Collatz conjecture. According to Collatz Conjecture, for any integer a

$$a_n = \begin{cases} \frac{1}{2} a_{n-1} & \text{for } a_{n-1} \text{ even} \\ 3\,a_{n-1} + 1 & \text{for } a_{n-1} \text{ odd} \end{cases}$$

Irrespective of the choice of n, the sequence will eventually converge to 1.

Sample Input: 6 Sample Output: 6, 3, 10, 5, 16, 8, 4, 2, 1	Sample Input: 13 Sample Output: 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

[7]

```
def Hailstone(n):
    if n == 1:
        print(1)
        return
    if n%2 != 0:
        print(n, end=(", "))
        return Hailstone(3*n+1)
    if n%2 == 0:
        print(n, end=(", "))
        return Hailstone(n/2)
```

Suppose you are given an object q of class Queue and a number k. Queue class supports all the basic operations like enqueue(n), dequeue() and peek() and has 2 attributes, front and size. **You do not need to implement the Queue class**. Your task is to rotate the q, k times.

Sample Input: [1, 2, 3, 4, 5, 6] 3 Sample Output: [4, 5, 6, 1, 2, 3]	Sample Input: ['a', 'b', 'c', 'd', 'e'] 2 Sample Output: ['c', 'd', 'e', 'a', 'b']

```
def rotate_queue(q,k):
    #your code goes here
```

Hint: enqueue(n) inserts n at the end of the queue. dequeue() removes the first element of the queue. You have to use both for this task. **[6]**

```
for i in range(k):
    temp = q.dequeue()
    q.enqueue(temp)

return q
```

4.
```
def change(l,i):
    if i==len(l):
        return l
    l[i] = 5*l[i]
    change(l,i+1)
    return l
a = [10,20,30,40]
l = change(a,0)
print('l = ',l,',a = ',a)
```

What will be the output?

a. l = [50, 100, 150, 200] ,a = [50, 100, 150, 200]
b. l = [50, 100, 150, 200] ,a = [10,20,30,40]
c. l = [10,20,30,40] ,a = [10,20,30,40]
d. l = [10,20,30,40] ,a = [50, 100, 150, 200]

[3]

5.
```
def change(l,i):
    if i==len(l):
        return l
    l[i] = 5*l[i]
    change(l,i+1)
    return l
a = [10,20,30,40]
l = change(a[:],0)
print('l = ',l,',a = ',a)
```

a. l = [50, 100, 150, 200] ,a = [50, 100, 150, 200]
b. l = [50, 100, 150, 200] ,a = [10,20,30,40]
c. l = [10,20,30,40] ,a = [10,20,30,40]
d. l = [10,20,30,40] ,a = [50, 100, 150, 200]

[3]

1. The array representation of a **binary search tree (BST)** is given below [None value means the node is empty]:

   [None, 6, 4, 11, 2, 5, 8, 12, None, None, None, None, None ,10, None, 20]

   (The first None value indicates a dummy node of the tree)

   Answer the following questions-

   A. Draw the BST. **[2.5]**

   B. A specific type of traversal prints out the node values in sorted order. What is the traversal's name? Write that particular traversal sequence of the tree in part A. **[2.5]**

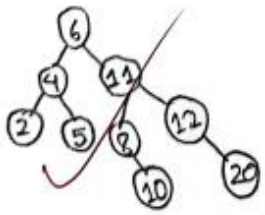
   C. Write the **post order traversal** sequence of the tree in part A. Use that traversal sequence to insert the elements in that order in an initially empty BST, and show the resulting BST. **[3]**

   Note: Consider the first element of the post order sequence as the root.

   D. Perform the following operations step by step on the Binary Search Tree you created in **part C**.

   i. Delete node 6 with the help of its successor.

   ii. Delete node 8 with the help of its predecessor. **[2]**



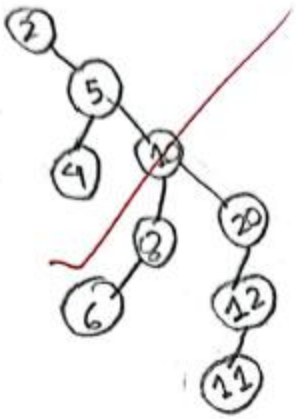
b) In order traversal.

2.5 Sequence: 2, 4, 5, 6, 8, 10, 11, 12, 20

c) Sequence: 2, 5, 4, 10, 8, 20, 12, 11, 6

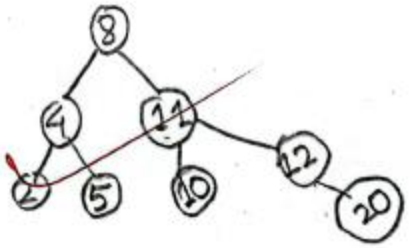
BST:

3/

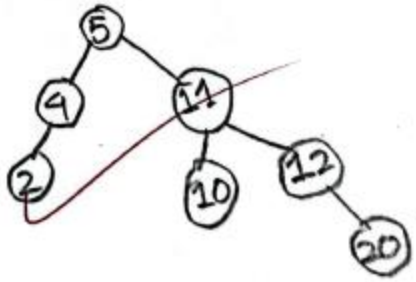


d) i) Deleting node 6:

2/



ii) Deleting node 8:



What is the maximum height of a tree with N nodes? Justify your answer with an example. [2]

The maximum height of a tree with N nodes is: count of nodes - 1.

Here we have 4 nodes and the height is 3.

2/ good

3. If we insert nodes into a BST in different orders, will it generate different binary trees? Justify your answer with examples. [3]

Yes, If we insert nodes into a BST in different orders, it will generate different binary trees.

From, 1 (a) we can see a binary tree inserted in a certain order. But in 1 (c), after changing the sequence the insertion also changed. and so the tree.

3

1. CO1 If a letter means enqueue, plus (+) means peek and an asterisk (*) means dequeue in a sequence, Draw the figure of the queue while you carry out the enqueue, dequeue and peek operations on the following sequence: 3

Q U * + * I * Z + I * I * + I *

Write down the dequeued and peeked values. Show your work elaborately.

2. CO4 Given an integer number, **write** a recursive function to count the number of even digits in that number. 7

```
def countEven(n) :
        #To Do
```

OR

```
public int countEven(int n){
    //To Do
}
```

3. CO4 Consider the following pseudocode: 5

```
printRecur(n):
    IF n > 9000:
      RETURN
    PRINT (n + 3)
    printRecur(3*n)
    PRINT (n + 3)
```

Simulate the above code to find the output of **printRecur(1000)**
[You must show your workings]

Question - 2

```
def countEven (n):
    if (n == 0):
        return 0
    else:
        r = n % 10
        if (r % 2 == 0):
            return 1 + countEven (n//10)
        else:
            return 0 + countEven (n//10)
```

Rough

1 2 3 4 5 6

7

Question - 1

Reading	queue	output
Q	Q	-
U	QU	Q
*	QU	QU
+	U	
*	-	QUU
I	I	QUU
*	-	QUUI
Z	Z	QUUI
+	Z	QUUIZ
I	ZI	QUUIZ

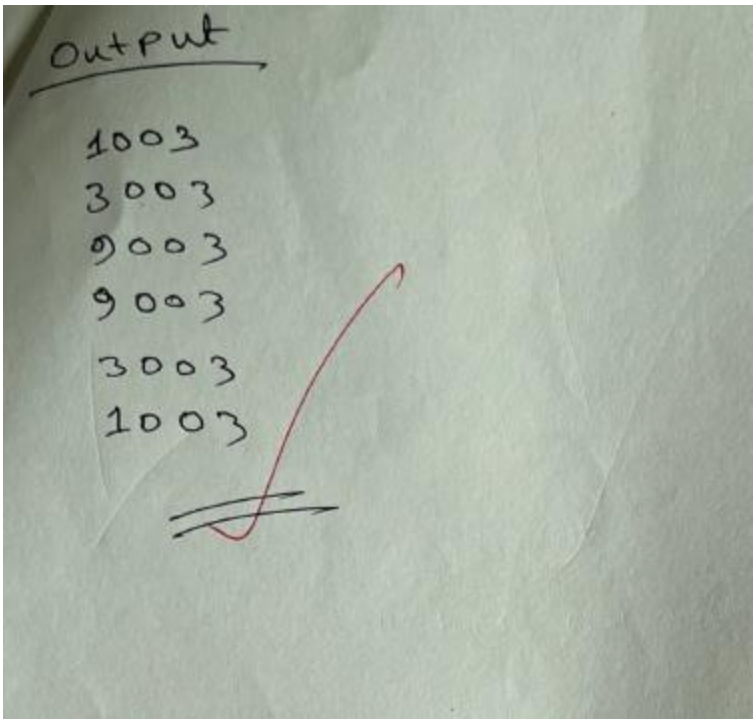
2.5

*	I	QUVIZZ
I	II	QUVIZZ
*	I	QUVIZZI
+	I	QUVIZZII
I	II	QUVIZZII
*	I	QUVIZZIII

Output : Q, U, V, I, Z, Z, I, I, I

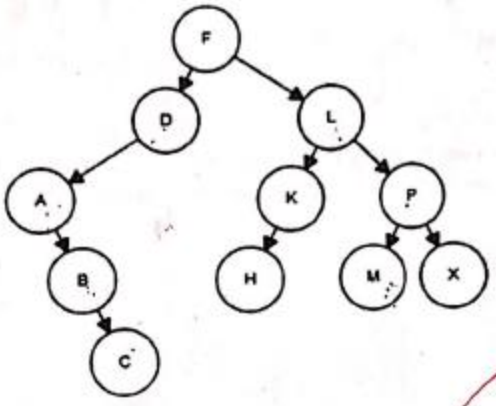
Question - 3

n	n > 9000	Remark.
1000	F	Print (1003) printRecun (3000) Print (1003)
3000	F	Print (3003) printRecun (9000) Print (3003)
9000	F	Print (9003) printRecun (27000) print (9003)
27000	T	Return


Output
1003
3003
9003
9003
3003
1003

1. Consider the given Binary Search Tree.

CO1
CO2



PLR
SRL

a. **What** is the successor of **C**? D [1]
b. **What** is the predecessor of **M**? L [1]
c. **Demonstrate** the following operations step by step on the given BST: [2]
   i. Delete node F
   ii. Delete node D
d. **Show** the elements of the Binary Search Tree in a sorted order. [You must show your work elaborately] [1]

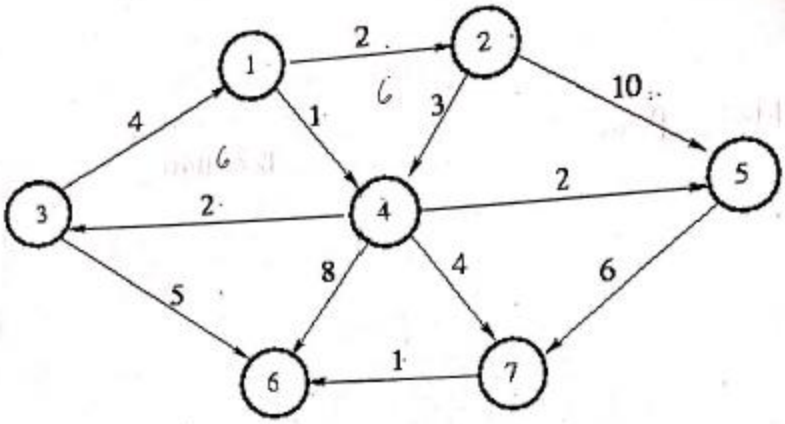
2. **Given the array representation of a binary tree [null value means the node is empty]:** 5

CO1
CO2

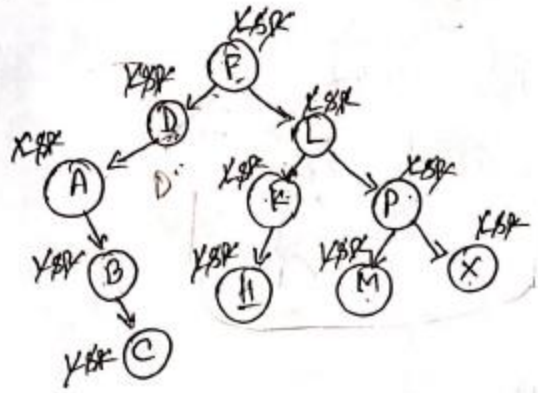
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
[null, 50, 17, 65, 12, null, 7, 86, 35, null, null, null, 9, 10, null, 28]

a. **Draw** the binary tree. [2]
b. **Write** the post-order traversal sequence of the tree. [1]
c. **Use** the post-order traversal sequence of **Question 2b** to insert the elements in that order in an initially-empty binary search tree, and **show** the resulting binary search tree. [2]

3. CO2 Consider the given Graph:



a. **Construct** the equivalent adjacency matrix representation of the graph. [3]
b. **Find** out the indegree and outdegree of each vertex of the graph. [2]



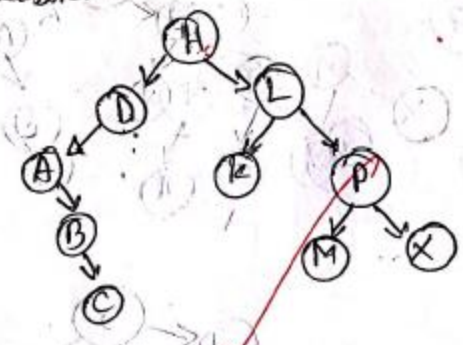
SRL
Right - left most
PLR

1

A, B, C, D, F, H, K, L, M, P, X
this is the in-order traversal of th tree and this will be the sorted order for this Binary Search Tree.

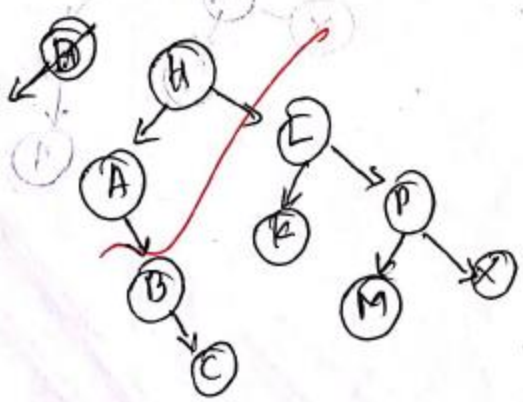
c) S-of F → H P of F? P of D → C

i) Deleting F with Successor H



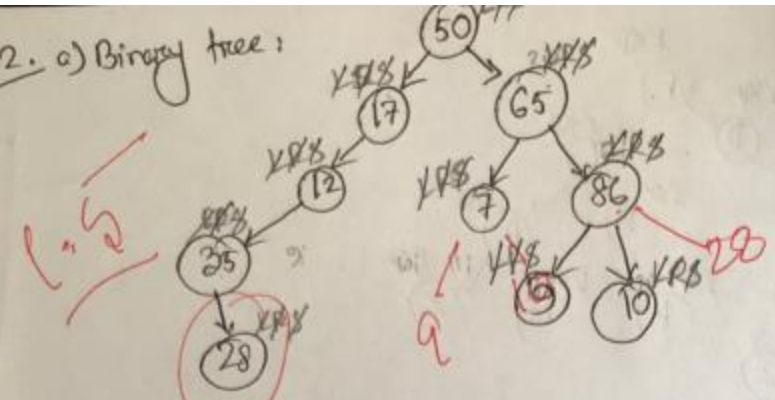
2

ii) Deleting D with ~~Successor H~~



1

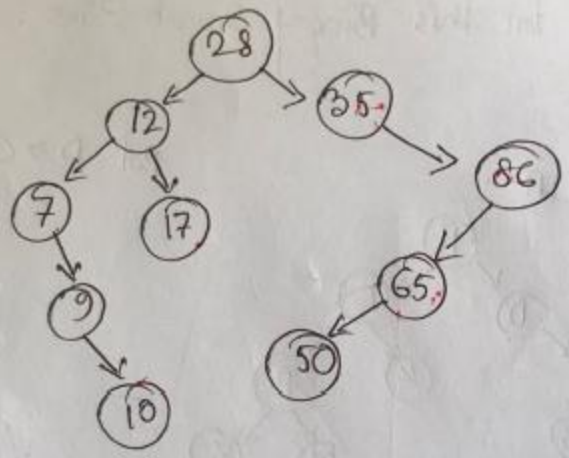
2. a) Binary tree:



b) Post order traversal:

28, 35, 12, 17, 7, 9, 10, 86, 65, 50

c)



	1	2	3	4	5	6	7
1	0	2	0	1	0	0	0
2	0	0	0	3	10	0	0
3	4	0	0	0	0	5	0
4	0	0	2	0	2	8	4
5	0	0	0	0	0	0	6
6	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0

3

b)

	in-degree	out-degree
1→	1	2
2→	1	2
3→	1	2
4→	2	4
5→	2	1
6→	3	0
7→	2	1

2