Rohan Nyati, 500075940, R177219148, Batch-5

Assignment -2

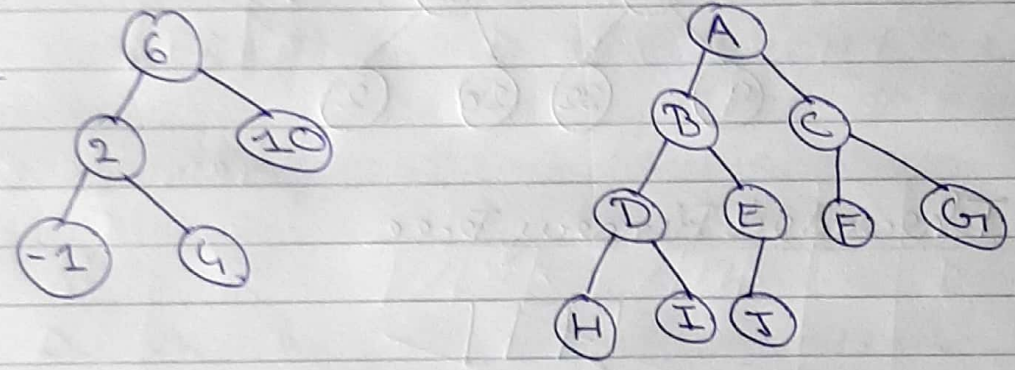
Date

Q.1 How is complete Tree and Perfect tree diff from each other? ~~Eg~~ Give Eg.

Ans **Complete Binary Tree :-**

Complete Binary Tree is a binary tree in which every level (except possibly the last) is completely filled, and all nodes are as far left as possible.

Eg.

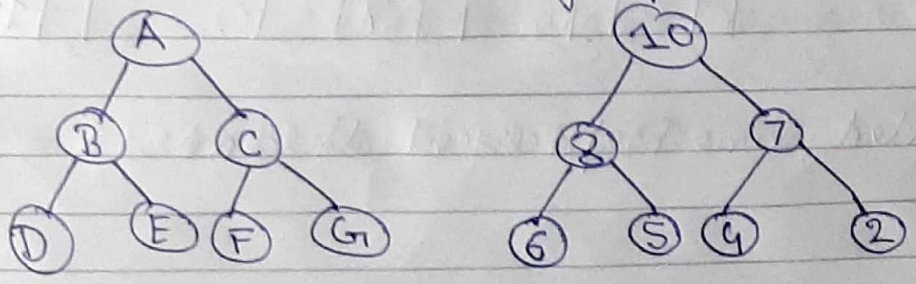


whereas,

Perfect Binary Tree

A Perfect Binary Tree is the one in which all leaves have the same depth or same level. In other words, every node has exactly two nodes and all levels are completely filled.

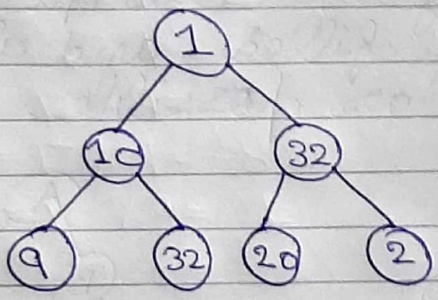
Eg.



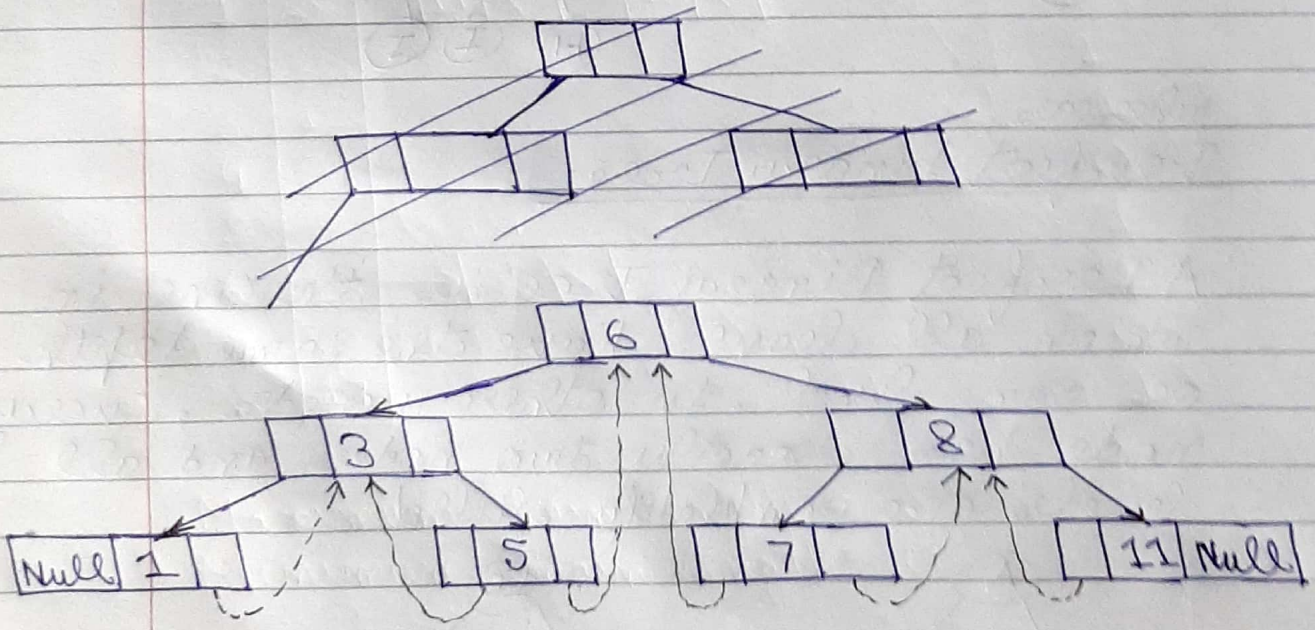
Date

Q.2] Construct a Binary tree with the breadth-first traversal order 1, 10, 13, 9, 32, 20
2. Create a threaded binary tree and state how the threaded binary tree is better than a Binary tree?

Ans. Binary tree with BFT order → 1, 10, 13, 9, 32, 20, 2



Threaded Binary Tree



Dotted line Represent threads

Date

The idea of Threaded Binary Tree is to make the traversal easy and without making the use of Stack or any Recursion. In this, all Right Null pointer are made to point to its inorder Successor and Left Null pointers are made to point inorder Predecessors. (Except the left and right most pointer which should always be Null.) The Threads are also useful in ~~Fas~~ Fast Accessing the Ancestores.

In Binary Tree all the Null pointer memory is wasted but in threaded Binary tree it makes the use of Null pointer to make the traversal of next or previous mode easier and faster.

Q.3) Consider the implementation of hash table using an array of 9 linked lists a seperate chaining method of collision avoidance.

(i) Insert the following 9 keys (5, 28, 19, 15, 20, 33, 8, 17, 10) in the given order in hash table using $h(k) = k \bmod 9$ as hash function and draw the final implementation neatly.

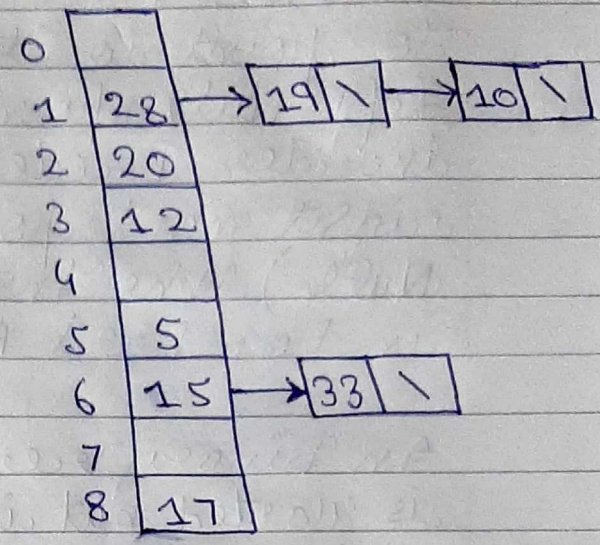
(ii) Calculate the maximum, minimum and average chain lengths in the hash table?

Date

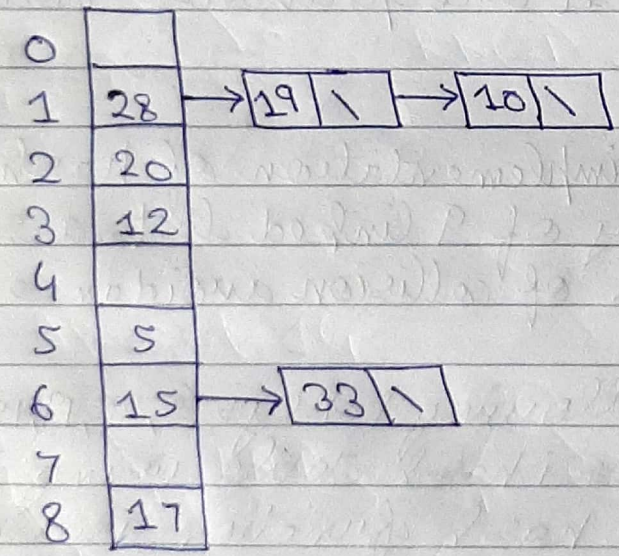
Ans (i) Keys = 5, 28, 19, 15, 20, 33, 12, 17, 10

$h(K) = K \bmod 9$

$h(5) = 5 \% 9 = 5$
$h(28) = 28 \% 9 = 1$
$h(19) = 19 \% 9 = 1$
$h(15) = 15 \% 9 = 6$
$h(20) = 20 \% 9 = 2$
$h(33) = 33 \% 9 = 6$
$h(12) = 12 \% 9 = 3$
$h(17) = 17 \% 9 = 8$
$h(10) = 10 \% 9 = 1$



(ii)



Chain Length
0
3
1
1
0
1
2
0
1

Maximum Chain Length = 3
Minimum Chain Length = 0

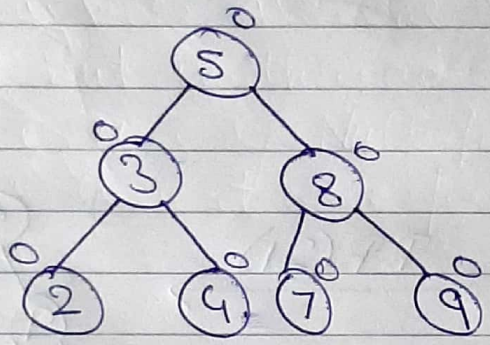
$$\text{Average} = \frac{0+3+1+1+0+1+2+0+1}{9} = \frac{9}{9} \Rightarrow \underline{1}$$

Date

Q.4] Explain the properties of an AVL tree. With suitable diagrams and explanations insert the elements 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3, 7 AVL tree by performing the necessary Rotations.

Ans. **Properties of AVL Tree**

(i) It must follow the properties of BST.
(ii) It has only balance factor 0 or -1 or 1, and it cannot be violated.

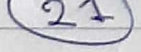


here Balance factor of all nodes is zero

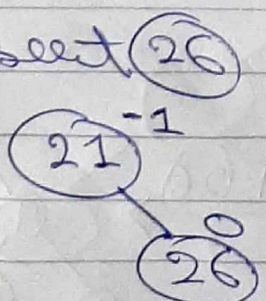
In this eg. we see that AVL tree also follows BST property i.e. left node always contains small element than root and right node contains greater element.

Keys = 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3, 7

Insert 21

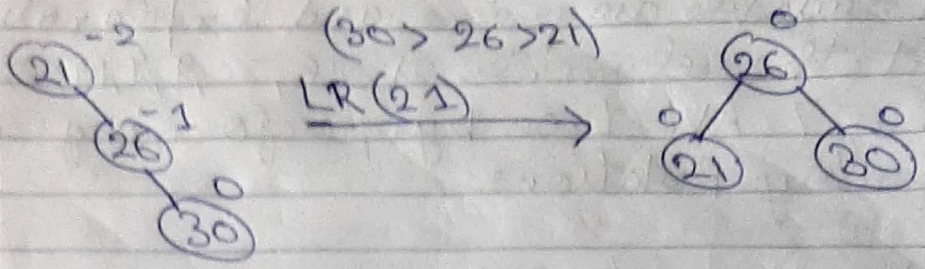


Insert 26 (26 > 21)

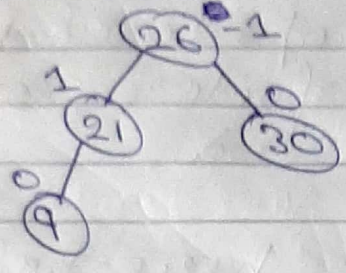


Date

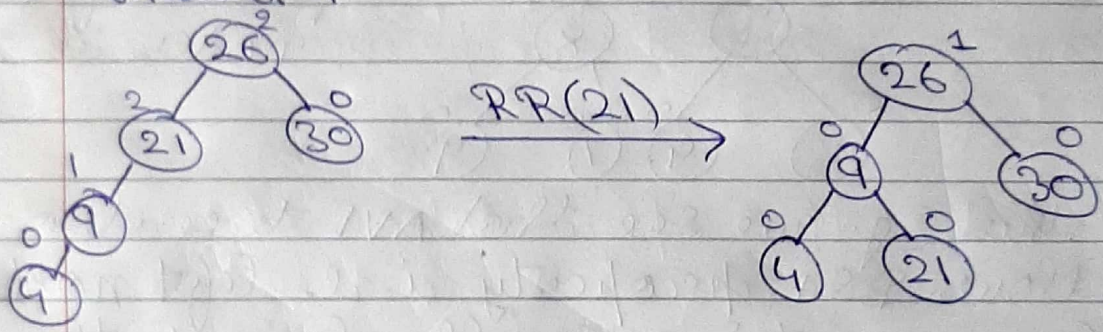
Insert 30



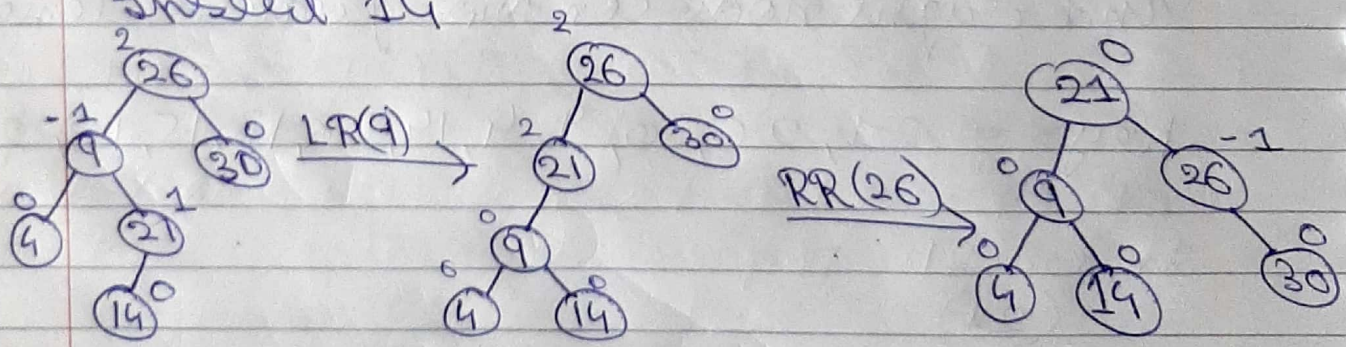
Insert 9



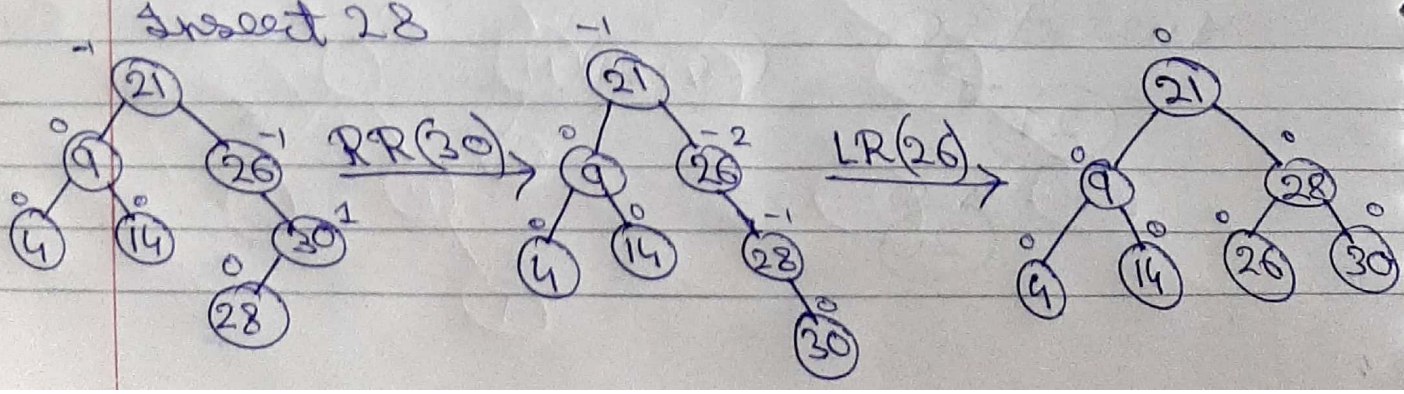
Insert 4



Insert 14

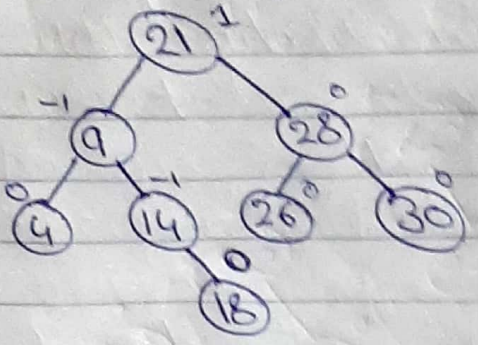


Insert 28

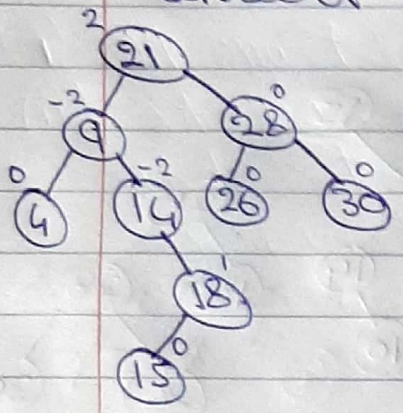


Date

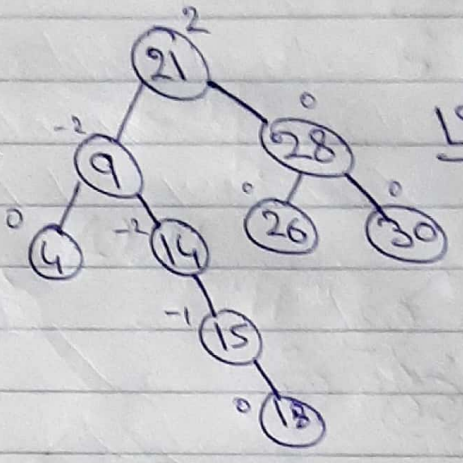
Insert 18



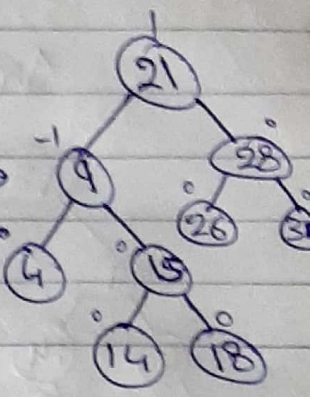
Insert 15



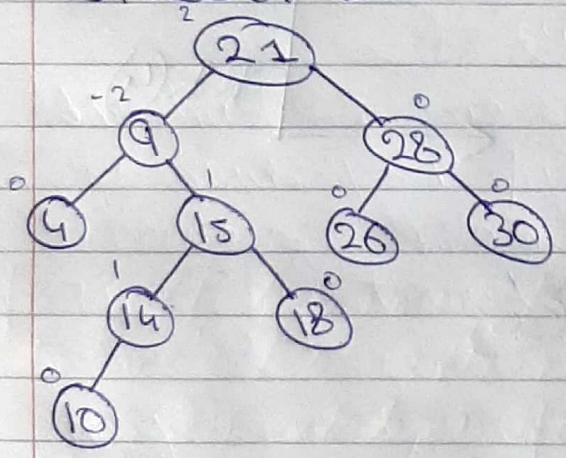
RR(18)



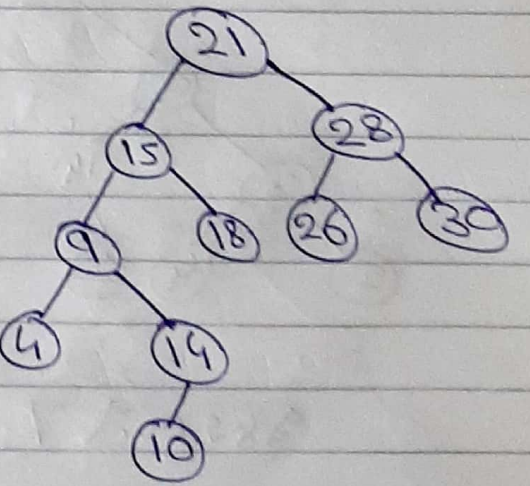
LR(15)



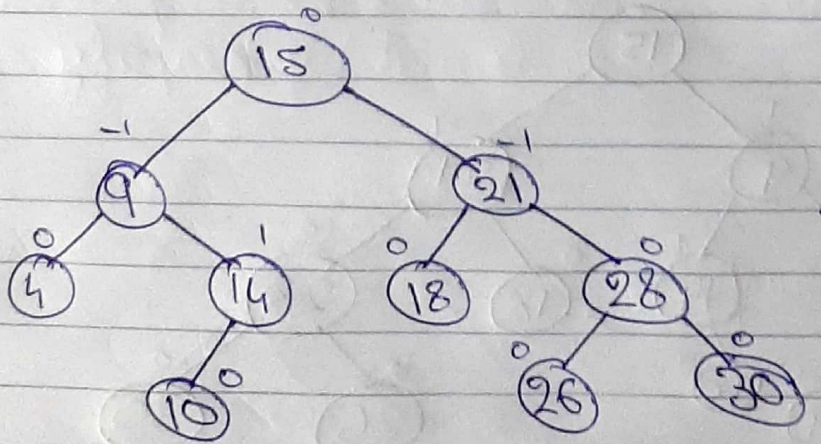
Insert 10



LR(9)

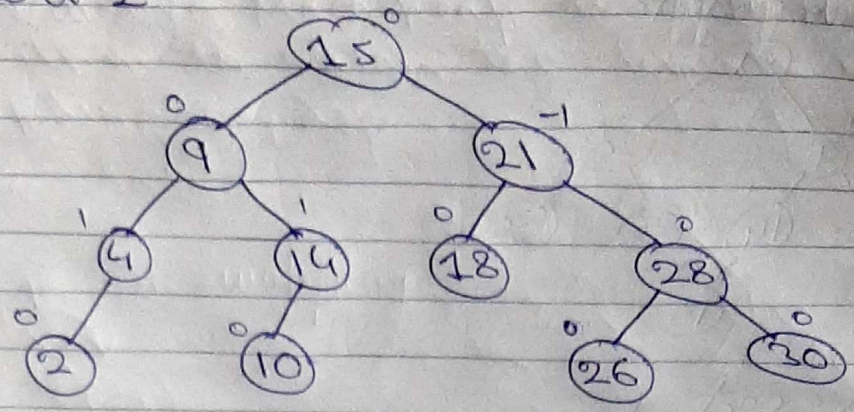


RR(21)

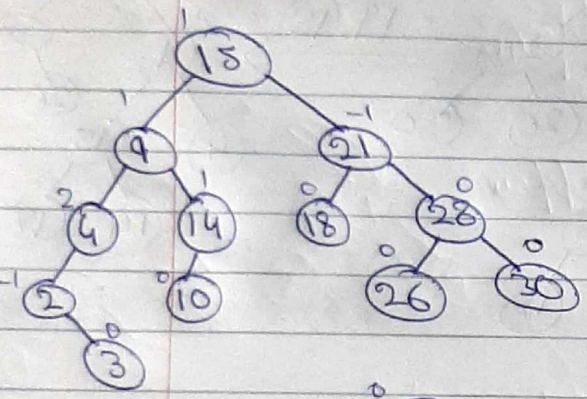


Date

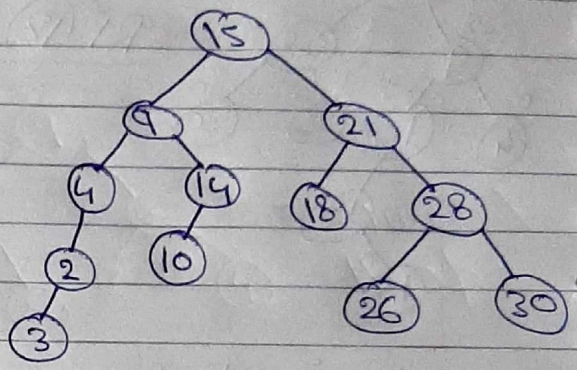
Insert 2



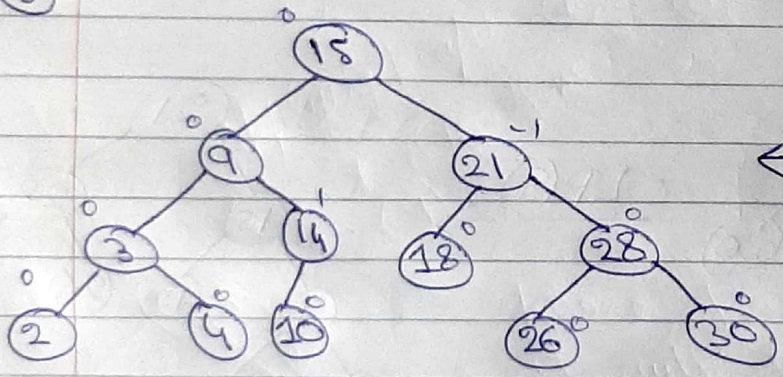
Insert 3



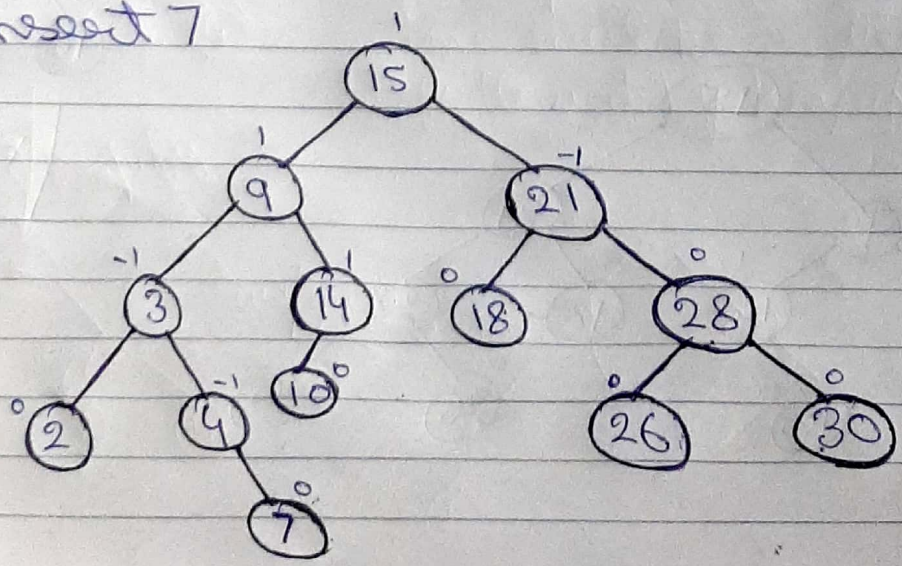
LR(2) →



RR(4) ←



Insert 7



Date

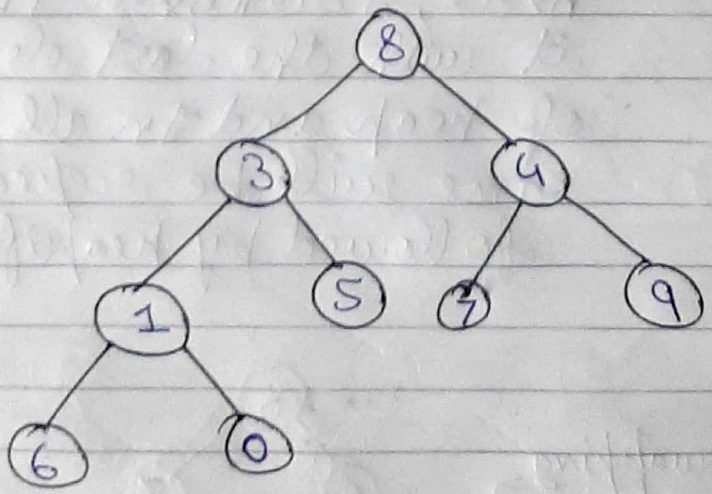
Q.5] Explain the properties of a Binary Heap tree. The Breadth-First Traversal pattern of a Complete Binary Tree (CBT) is 8, 3, 4, 1, 5, 7, 9, 2, 6, 0. Construct a CBT and later sort the elements in the CBT using Heap Sort Algorithm to obtain the elements in Descending order.

Ans. **Properties of Binary Heap tree**

(i) **Ordering** → Nodes must be arranged in an order according to values. The values should follow min-heap or max-heap property.

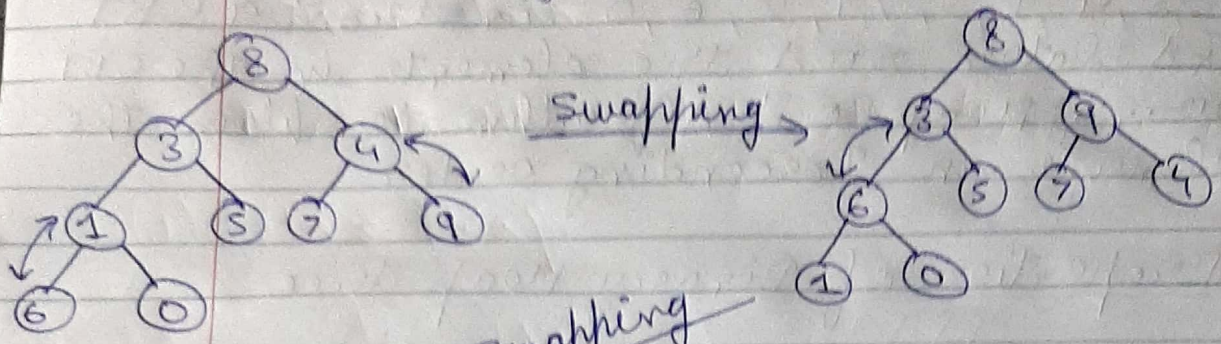
(ii) **Structural** → All levels in a heap should be full. We can say that it should be a complete Binary tree.

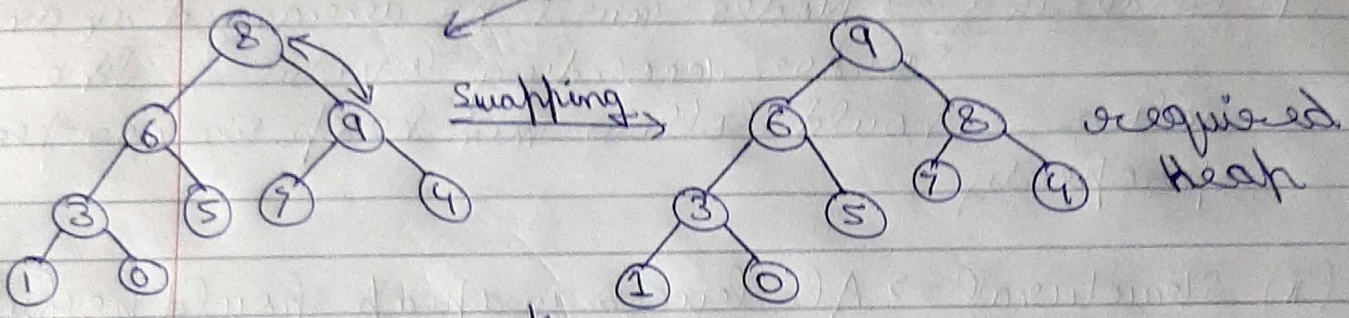
CBT according to BFT pattern



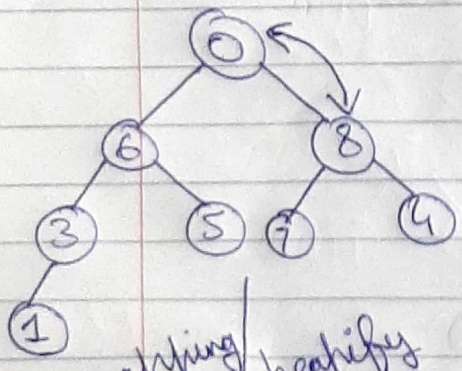
Date

Constructing Max heap to apply HeapSort algorithm to obtain the elements in descending order.

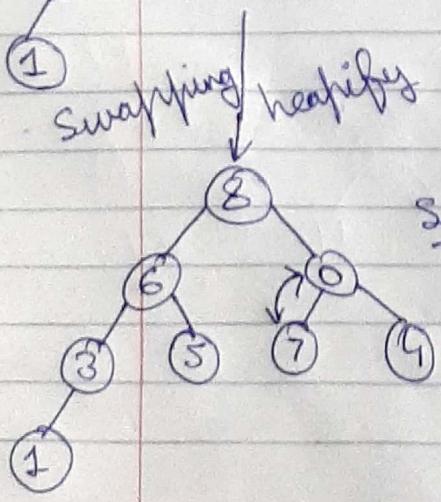




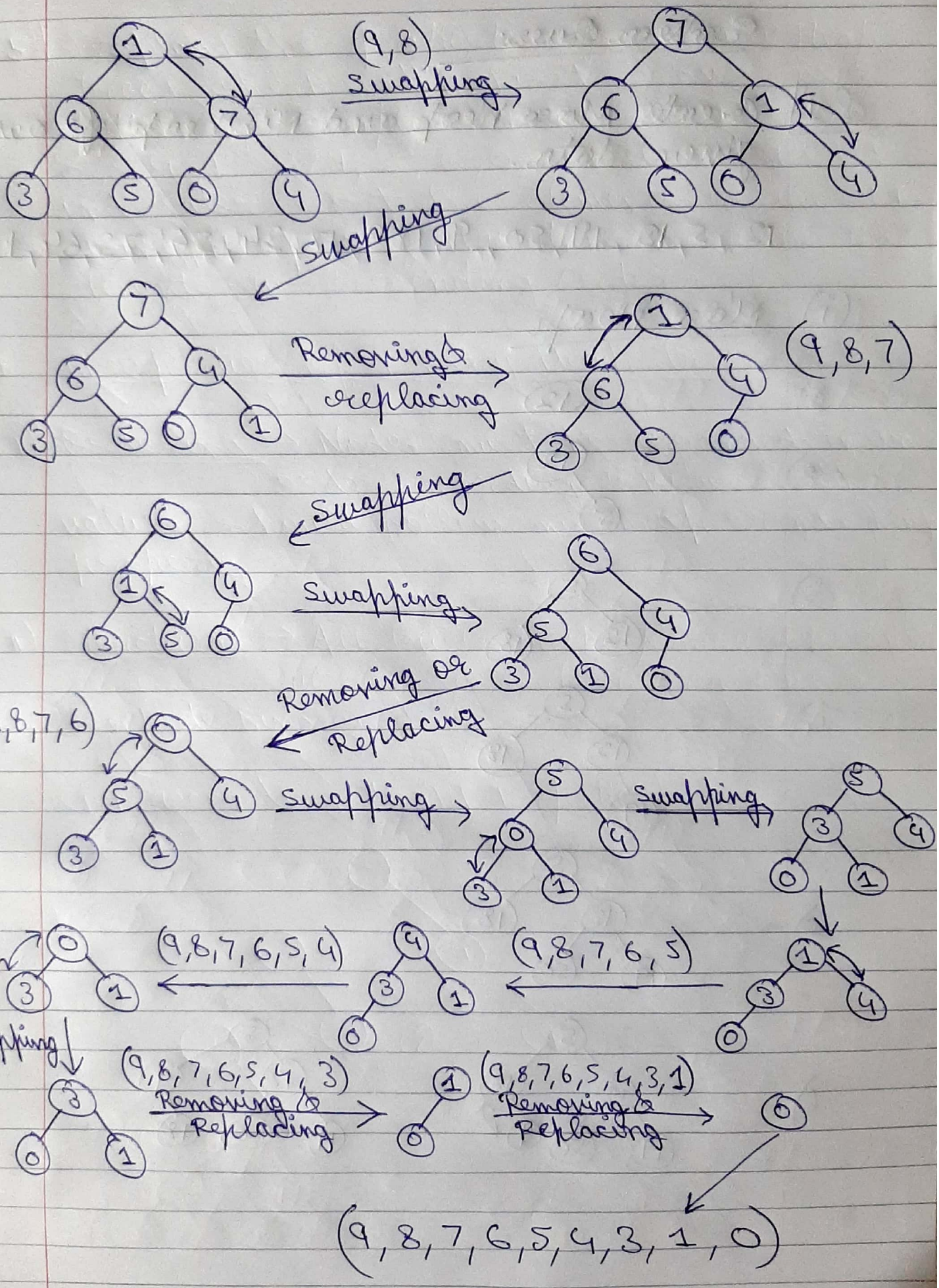
Applying heap sort algorithm (9)



here, 9 removed 9 by replacing it with the last element of heap and in all cases same will be repeated followed by heapify.



Date



Date

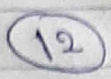
Extra Ques:

8.1) Create Max Heap and Min Heap from given data.

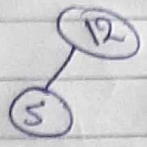
12, 5, 18, 19, 50, 9, 183, 17, 34, 54, 73, 64, 10

(i) Max Heap

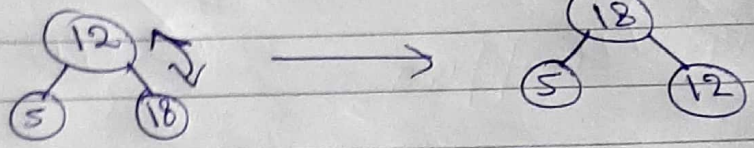
Insert 12



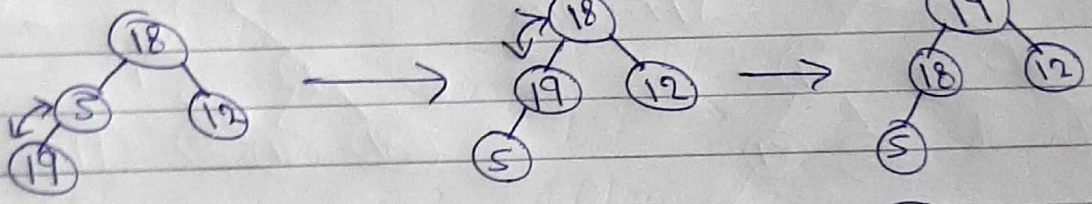
Insert 5



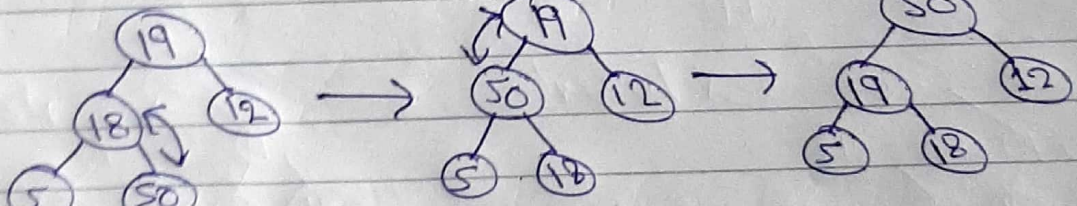
Insert 18



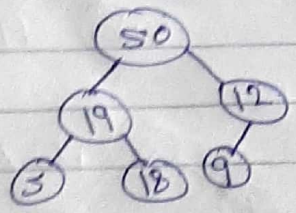
Insert 19



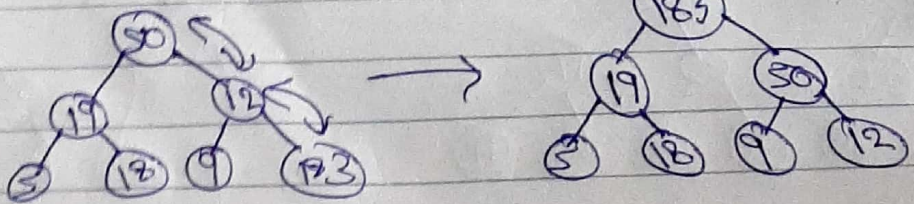
Insert 50



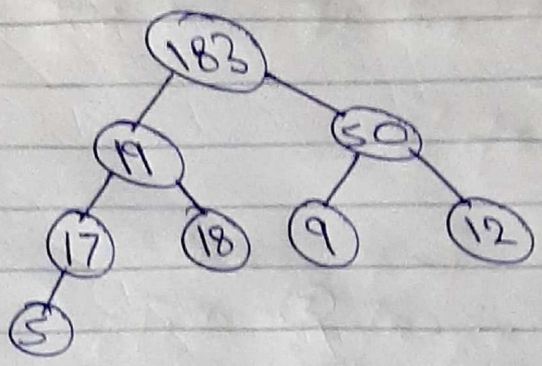
Insert 9



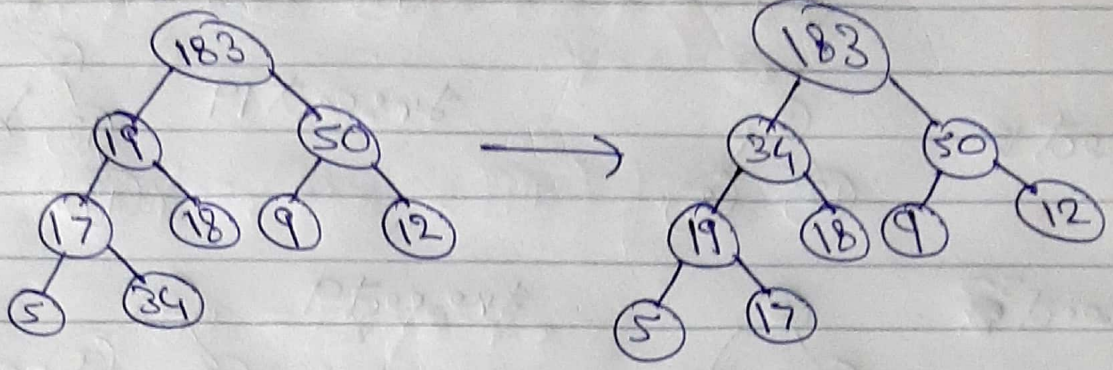
Insert 183



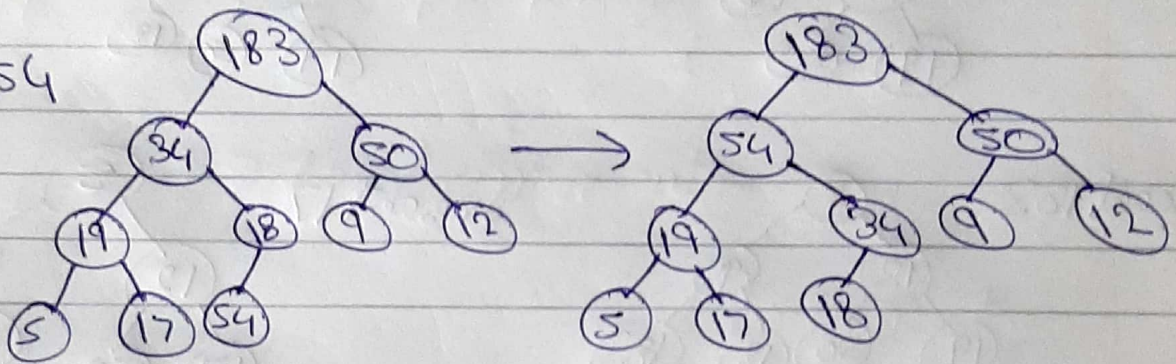
Insert 17



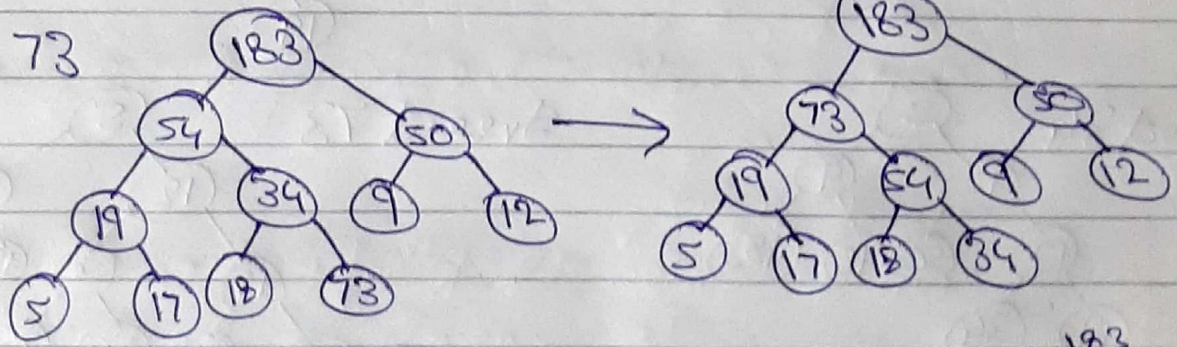
Insert 34



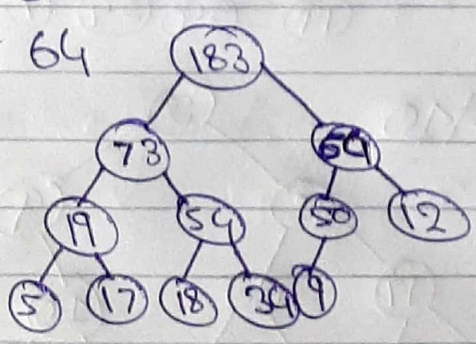
Insert 54



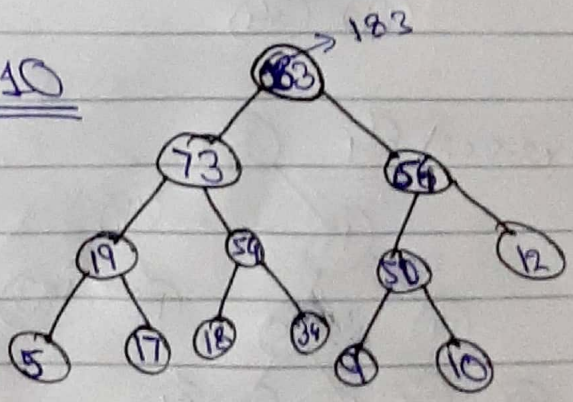
Insert 73



Insert 64



Insert 10

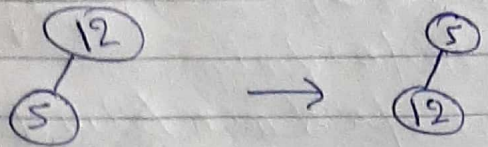


Date

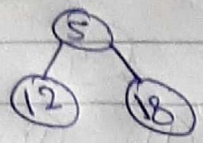
(ii) Min Heap

Insert 12

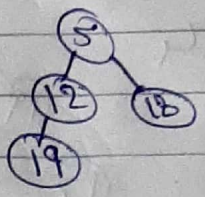
Insert 5



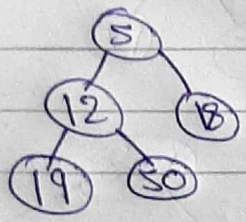
Insert 18



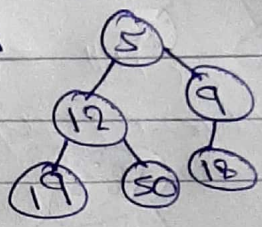
Insert 19



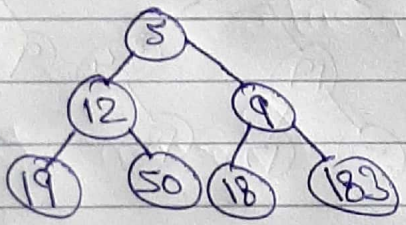
Insert 50



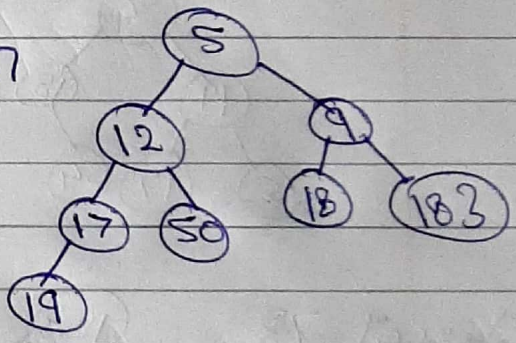
Insert 9



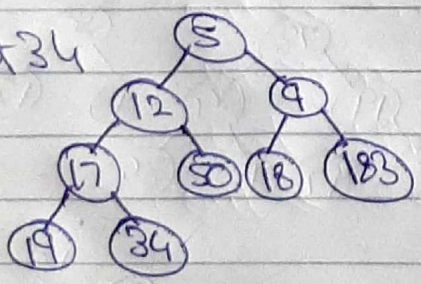
Insert 183



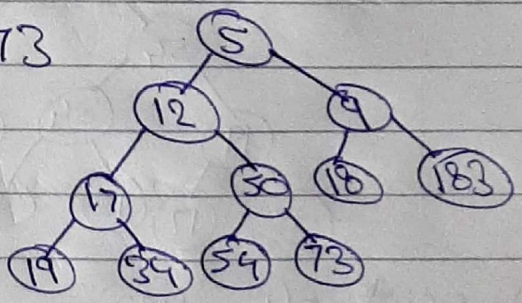
Insert 17



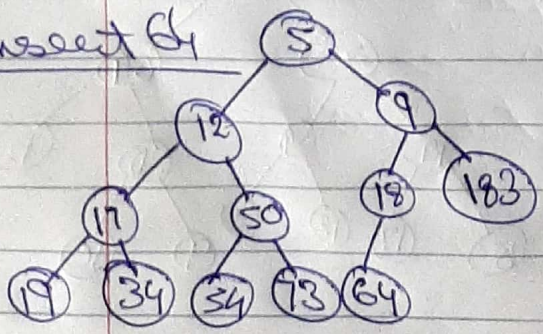
Insert 34



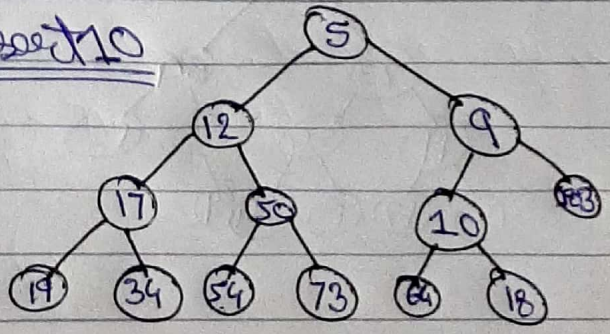
Insert 73



Insert 64



Insert 10



Date

Q.2] Write code to find k^{th} largest element in an array through Heap Sort.

```
#include<stdio.h>
#include<iostream>
using namespace std;
void swap(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}
void minheapify(int a[], int size, int i)
{
    int l = 2*i;
    int r = 2*i + 1;
    if (l < size && a[l] < a[smallest])
        smallest = l;
    if (r < size && a[r] < a[smallest])
        smallest = r;
    if (smallest != r)
    {
        swap(&a[i], &a[smallest]);
        minheapify(a, size, smallest);
    }
}

void buildminheap(int a[], int size)
{
    for (int i = size/2; i >= 0; i--)
        minheapify(a, size, i);
}
```

Date

```
int Kth largest (int a[], int size, int K)
{
    int minheap[K];
    int i;
    for(i=0; i<K; i++)
        minheap[i] = a[i];
    build minheap (minheap, K);
    for(i=K; i<size; i++)
    {
        if (a[i] > minheap[0])
        {
            minheap[0] = a[i];
            minheapify (minheap, K, 0);
        }
    }
    return minheap[0];
}
int main()
{
    int K, n;
    cout << "enter the size of Array" << endl;
    cin >> n;
    int a[n];
    cout << "enter the Kth position of element
            to search in the array: " << endl;
    cin >> K;
    cout << "enter the elements of array:" << endl;
    for(int j=0; j<n; j++)
    {
        cin >> a[j];
    }
```

Date

```
int size = sizeof(a) / sizeof a[0];
cout << " " << kth largest(a, size, k) << endl;
return 0;
}
```

Output:

```
Enter the Size of Array:
 5
Enter the Kth position of element to
search in the array: 2
Enter the elements of the Array:
 5
 24
 6
 8
 13
The Kth ~~element~~ largest element is: 13
```