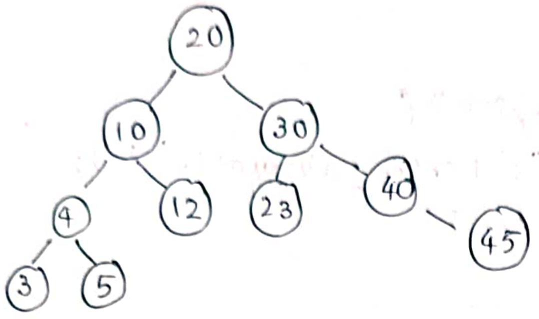
# ADSJ ASSIGNMENT

21B81A6754

1) What is Sorted Chain? Implement dictionary using Sorted Chain.

2) Delete the keys 12, 23, 30, 45.



3) Explain rotations in AVL tree.

4) What is Heap Sort? Sort the following keys using heap sort technique?

0, 30. 46, 27, 3, 11, 19, 21, 42, 9, 33

## Answers

1) A sorted chain is a data structure that combines the features of a dictionary and a linked list.

It is a dictionary where the key-value pairs are stored in a sorted order and based on the keys. It allows for efficient searching, insertion, and deletion operations, as well as iterating over the keys in a sorted order.

To implement dictionary using Sorted Chain, you use TreeMap class, which is a Red-Black tree based on the implementation of SortedMapInterface.

It sorts the value according to the comparator.

Code:

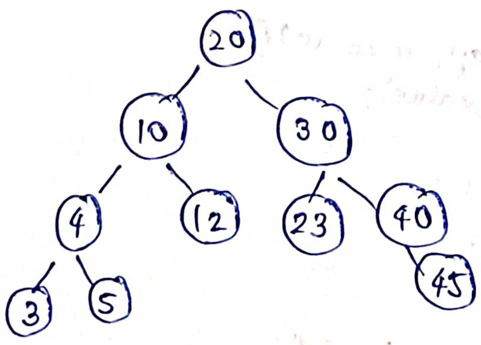
```
import java.util.*;
public class SortedChainDictionary<K extends Comparable<K,V> {
        privateTreeMap<K,V>map;
        public SortedChainDictionary(){
                map = new TreeMap<>();
        }
        public void put(K key, V value){
                map.put(key,value);
        }
        public V get(K key){
        return map.get(key);
        }
        public void remove(K key){
                map.remove(key);
```

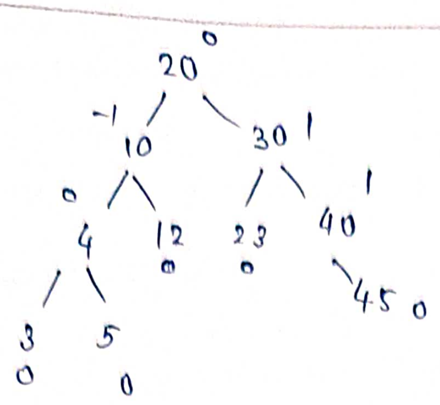
```
    }
public Set< K> KeySet(){
        return map.keySet();
    }
public void printDictionary(){
        for (Map.Entry<K,V> entry: map.entrySet()) {
                Sopln("("+entry.getKey()+":"+ entry.getValue()+")");
        }
    }
public static void main(String[]args){
    SortedChainDictionary <Integer, String> myDict = new SortedChainDictionary<>();
    myDict.put(5,"five");
      "         ( 3,"three");
      "         ( 8,"eight");
      "         ( 1,"one");

    Sopln(" Dictionary : ");
    myDict.printDictionary();
    myDict.remove(3);
    Sopln("\nDictionary after removing 3");
    myDict.printDictionary();
```

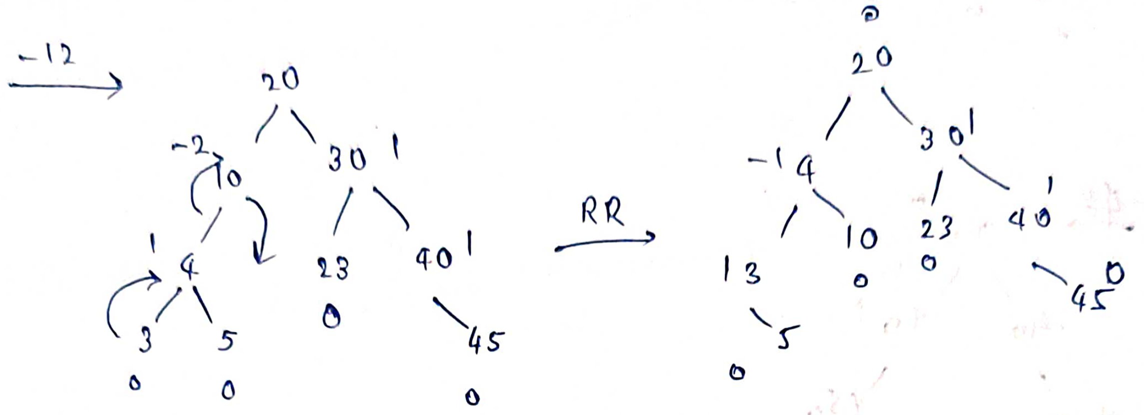
O/P -

```
Dictionary
 (1:one)
 (3:tree)
 (5:five)
 (8:eight)
Dictionary after removing key 3:
  (1:one)
  (5:five)
  (8:eight)
```

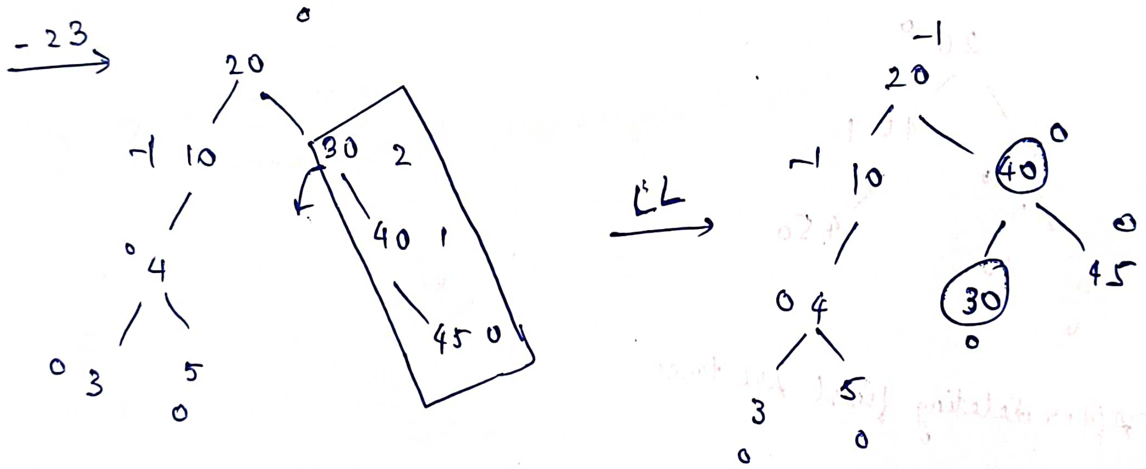
2)





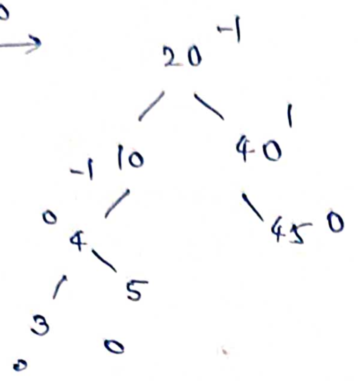


-12
RR

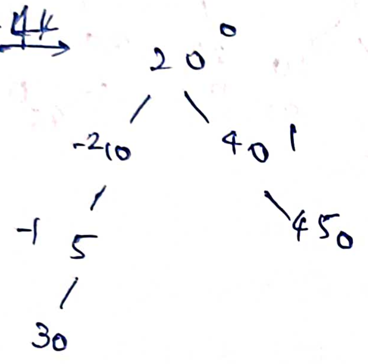


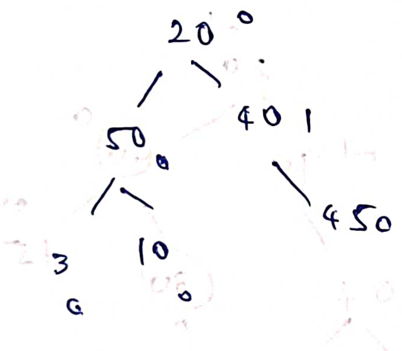
-23
LL

$\xrightarrow{-30}$

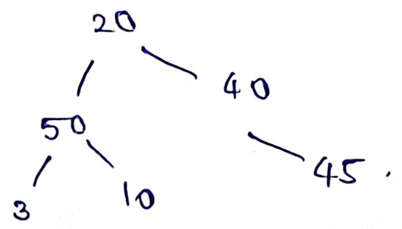


$\xrightarrow{-44}$





After deleting final AVL tree



Rotations are fundamental operations used in AVL trees to maintain their balance after insertions (or) deletions.

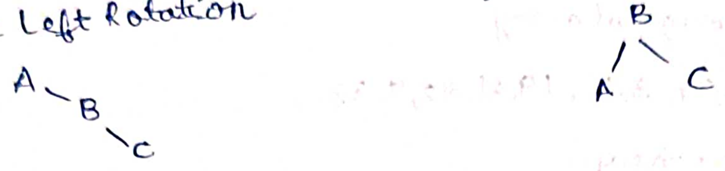
There are 4 types of rotations commonly used in AVL trees:

1. Left Rotation (LL Rotation):

A left rotation is performed to restore the balance when the right subtree of a node becomes taller than the left subtree of a node becomes taller than the left subtree by more than one level.

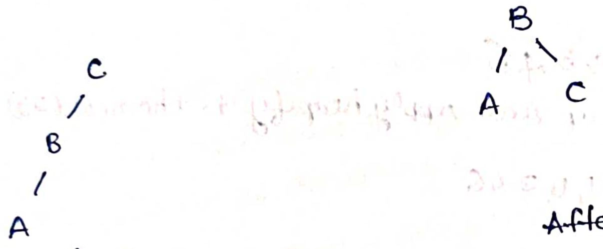
Before Left Rotation

After Left Rotation



2. Right Rotation (RR) Rotation-

It restores the balance when left subtree of a node becomes taller than the right subtree by more than one level.

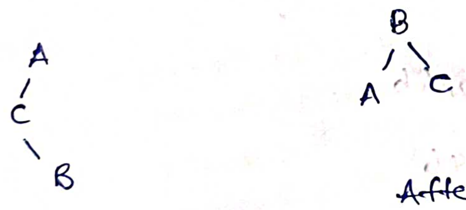


Before

After.

3. Left-Right (LR) Rotation-

It is a combination of left and right rotations.

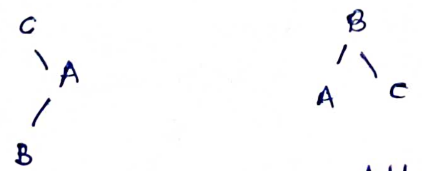


After.

Before

4. Right-Left (RL) Rotation-

It is a combination of right and left rotations.



After.

Before

4) Heap Sort -

It is a comparison based sorting algorithm that uses the concept of a binary heap data structure to efficiently sort an array or list of elements.

Heap Sort Algorithm consists of two main steps:

1. Building a max-heap
2. Sorting

1. Build the max-heap

Start with original array

0, 30, 46, 27, 3, 11, 19, 21, 42, 7, 33

Convert into max-heap.

46, 30, 42, 27, 9, 11, 19, 21, 0, 3, 33

2. Sorting:

Swap the maximum element (46) with the last element (33).

33, 30, 42, 27, 9, 11, 19, 21, 0, 3, 46

Reduce the size of the heap and apply heapify to the root (33):

42, 30, 33, 27, 9, 11, 19, 21, 0, 3, 46

Repeat the process:

30, 27, 33, 21, 9, 11, 19, 3, 0, 42, 46

Next,

27, 21, 33, 3, 9, 11, 19, 0, 30, 42, 46

Next,

11, 9, 19, 3, 0, 21, 33, 27, 30, 42, 46

9, 3, 19, 0, 11, 21, 33, 27, 30, 42, 46

0, 3, 19, 9, 11, 21, 33, 27, 30, 42, 46

Finally the sorted array is -

0, 3, 9, 11, 19, 21, 27, 30, 33, 42, 46.