Home Assignment - I

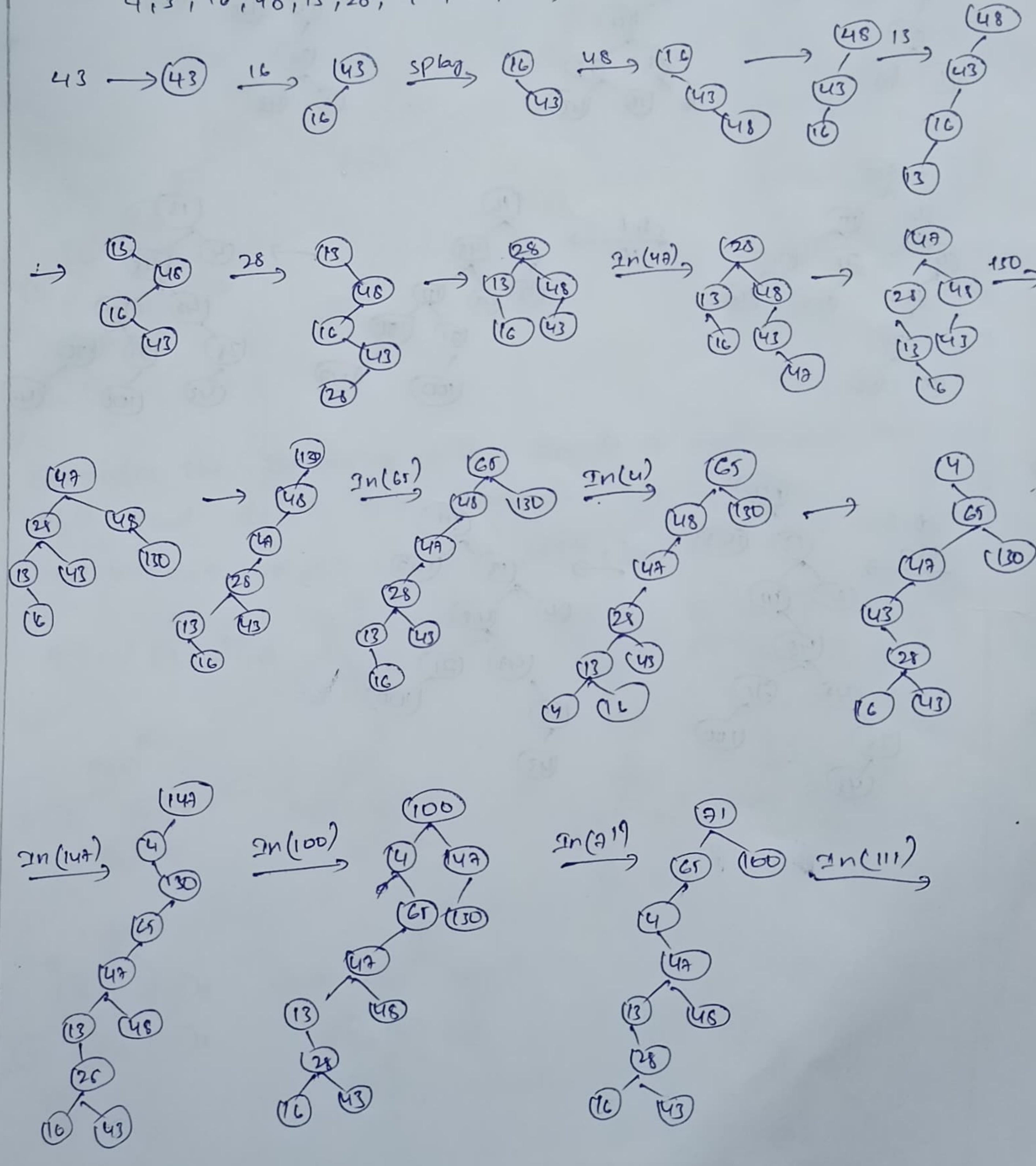
S. Venu
208W1A12C3

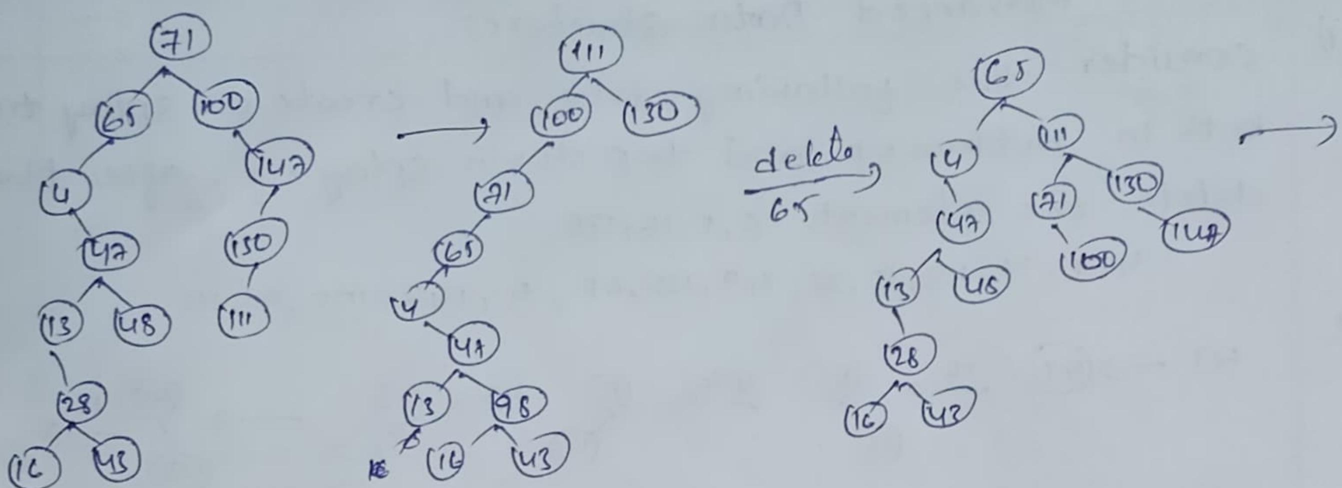
Advanced Data Structures

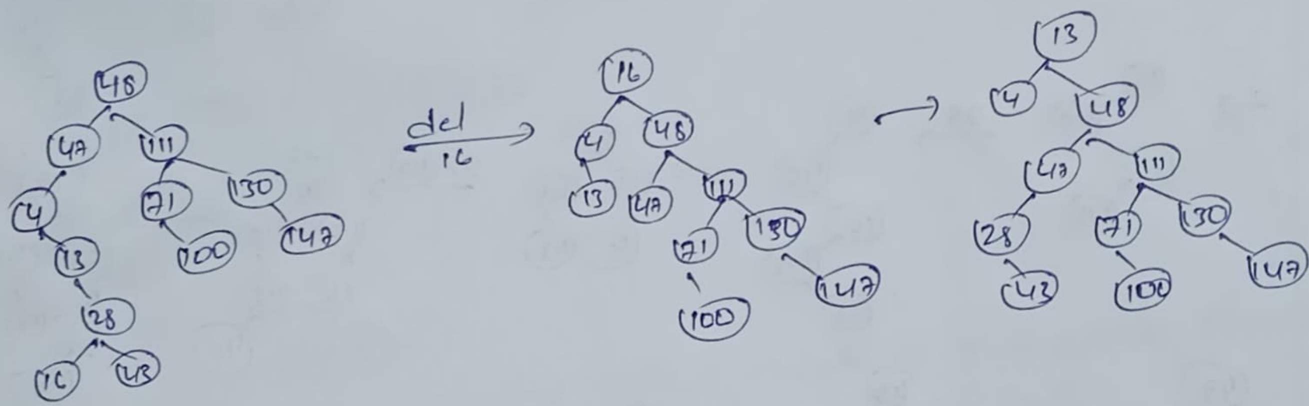
1) Consider the following data and create a splay tree both in bottom up and top down splaying. After that delete the elements 6, 5, 16, 130.

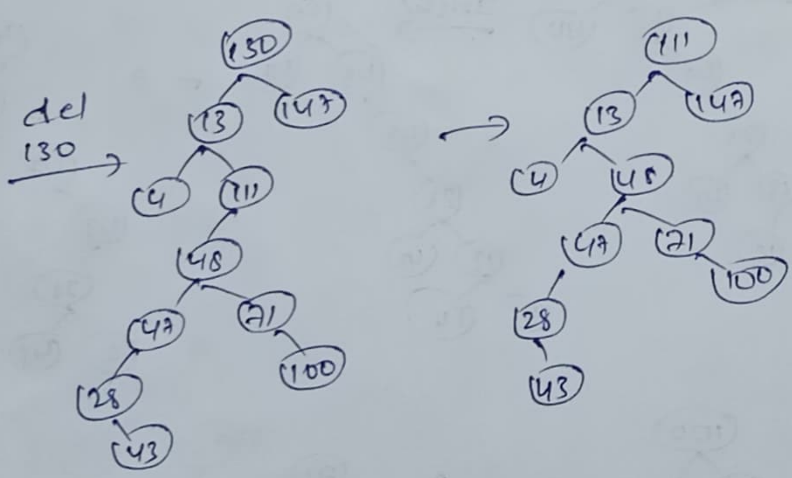
4, 3, 16, 48, 13, 28, 47, 130, 65, 4, 147, 100, 71, 111

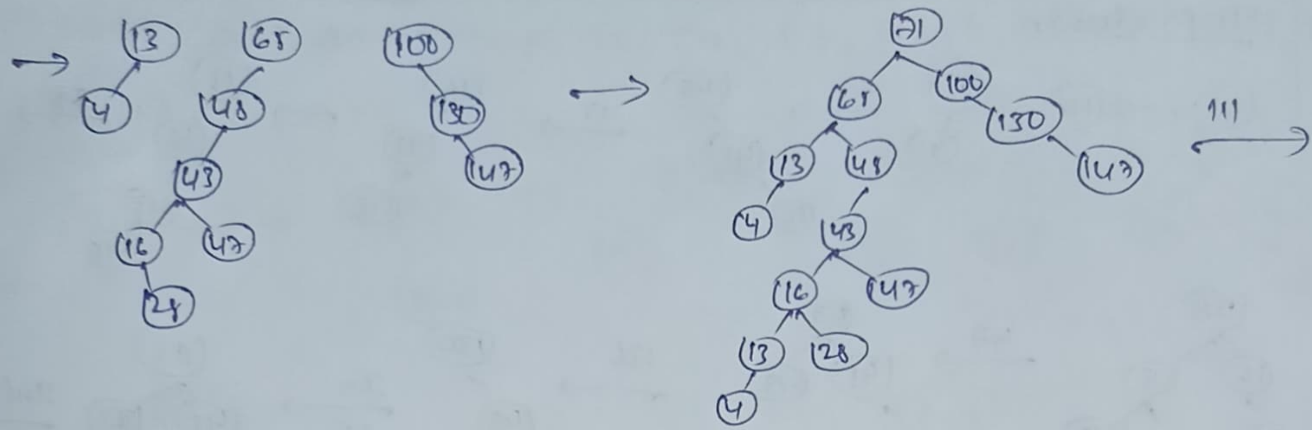
Sol

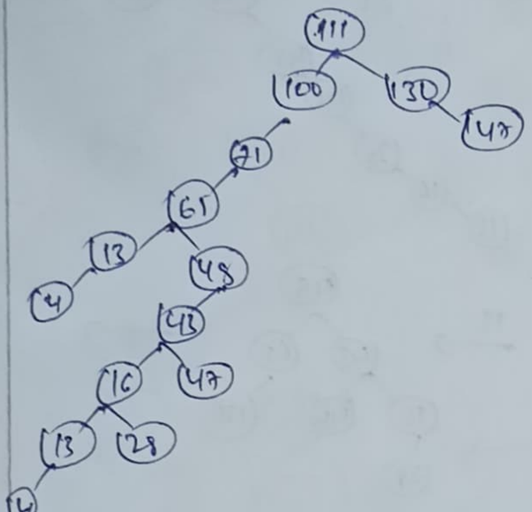



delete
65


del
16


del
130

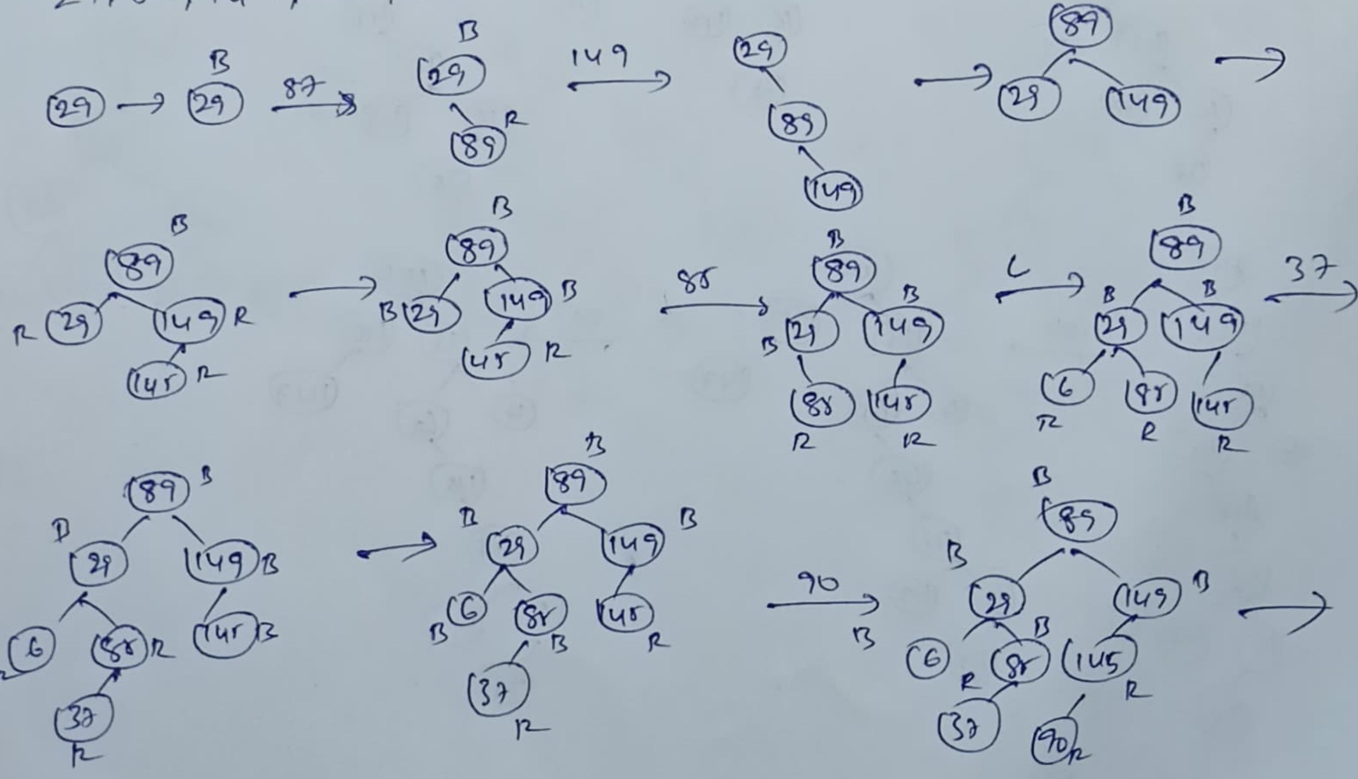




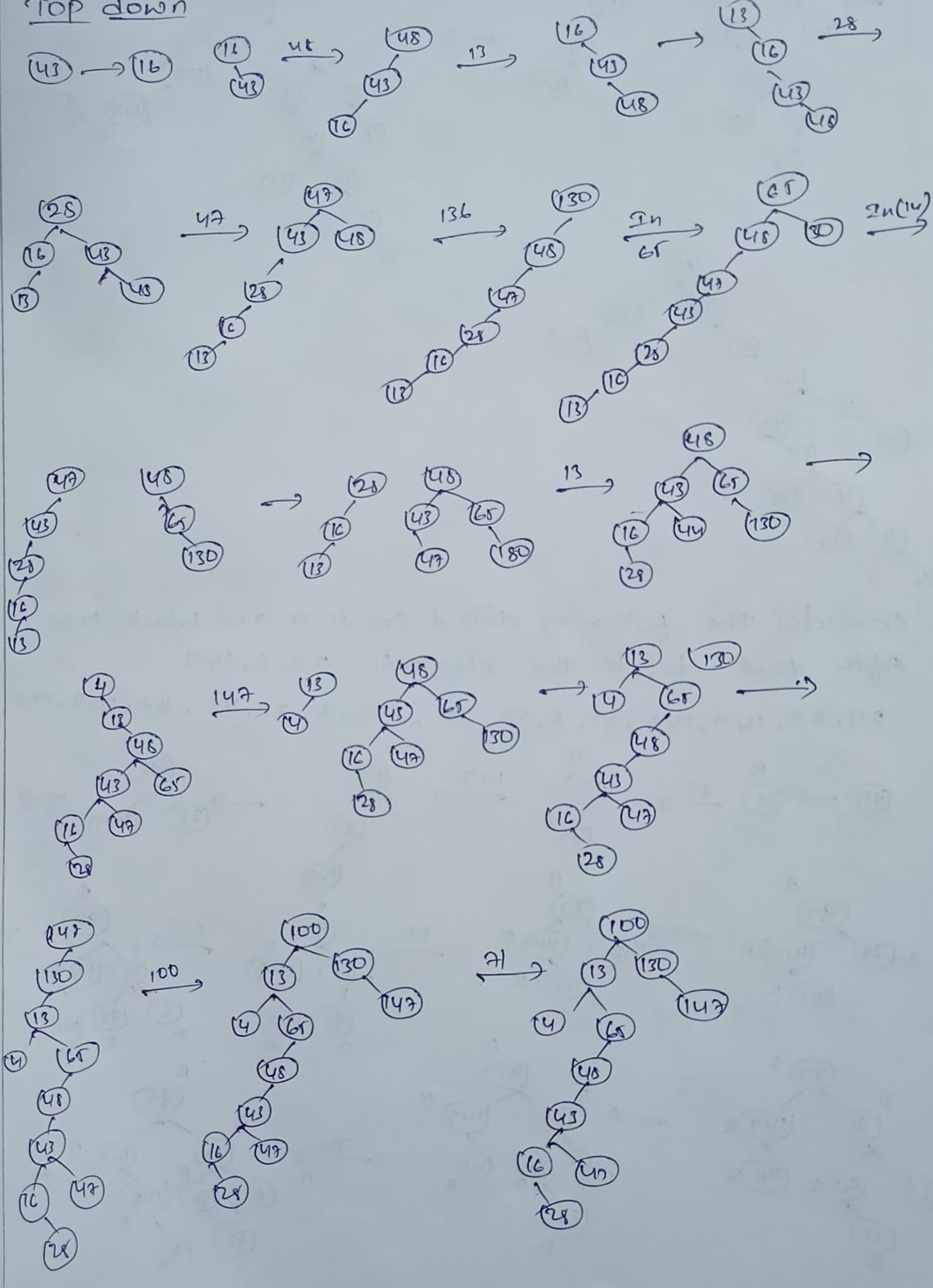
2) Consider the following data & create a red black tree.
After that delete the elements 29, 6, 149

29, 89, 149, 148, 85, 6, 37, 90, 136, 6, 33, 42, 87, 107, 93

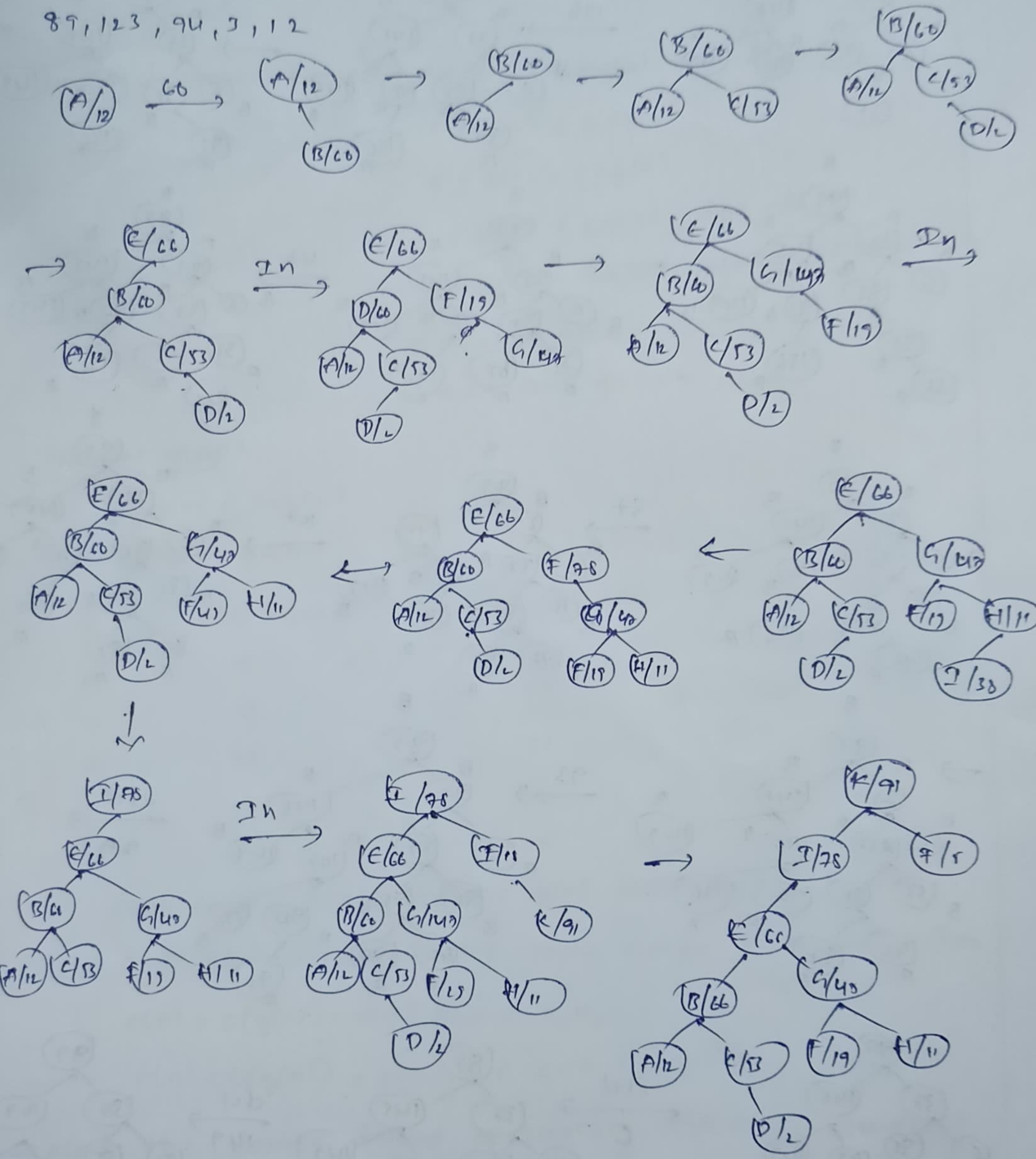
Sol:

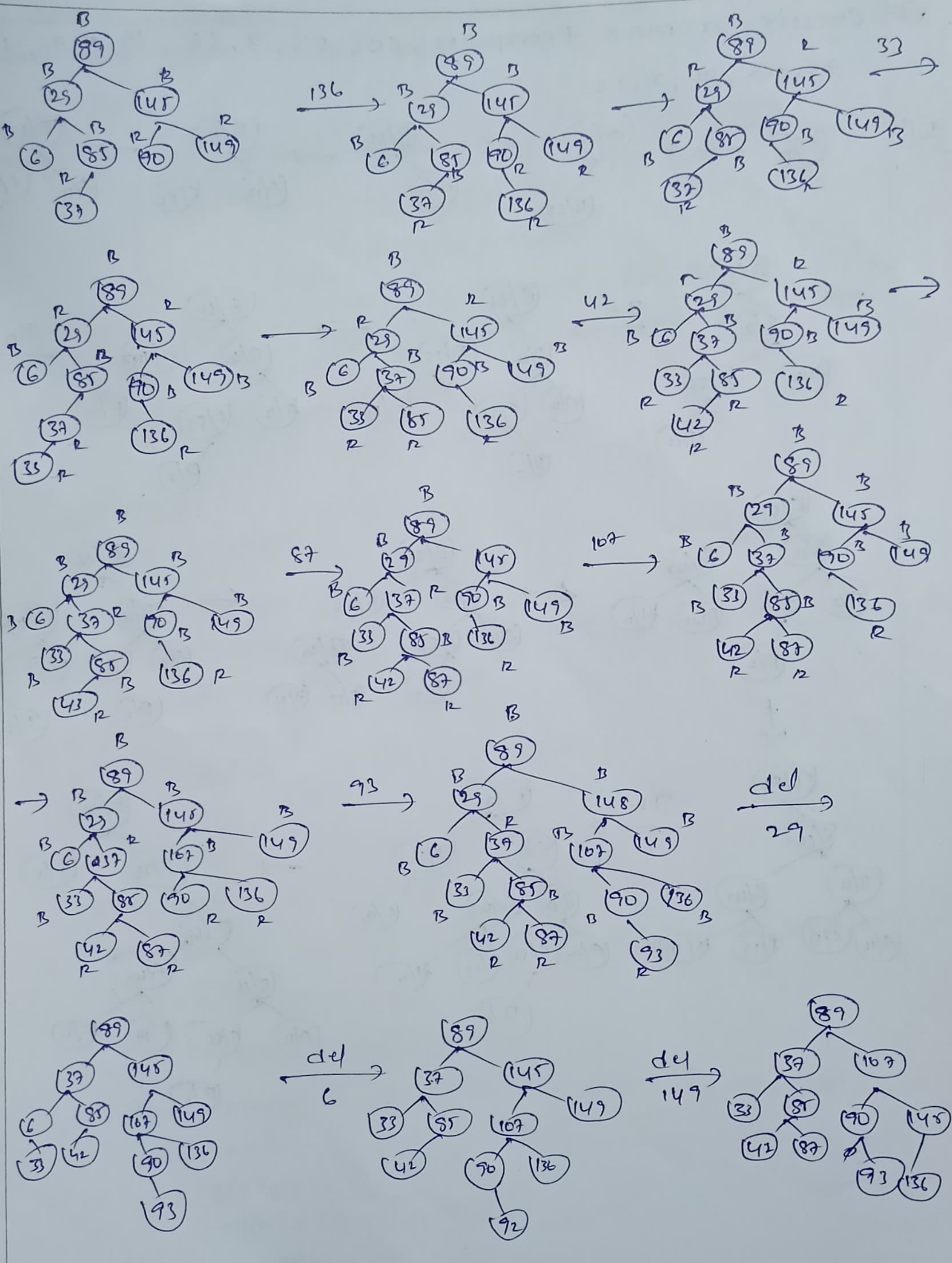


Top down



Create a max treap 12, 60, 53, 8, 66, 19, 147, 11, 78, 89, 123, 94, 5, 12




136
33
42
87
107
93
del 29
del 6
del 149

) Find the upper bound of running time of lineal function

$F(n) = 6n+3$

To find upper bound $0 \le F(n) \le c \times g(n)$ for all $n \ge n_0$

$0 \le F(n) \le c \times g(n)$

$0 \le 6n+3 \le c \times g(n)$

$0 \le 6n+3 \le 6n+3n$, for all $n \ge 1$

(there can be such infinite possibilities)

$0 \le 6n+3 \le 9n$

so $c = 9$ and $g(n) = n$, $n_0 \ge 1$

Tabular graph

$0 \le 6n+3 \le c \times g(n)$

$0 \le 6n+3 \le 7n$

n	$F(n) = 6n+3$	$c \cdot g(n) = 7n$
1	9	7
2	15	14
3	21	21
4	27	28
5	33	35

From table, $n \ge 3$, $F(n) \le c \times g(n)$ holds true

$c = 7$, $g(n) = n$ and $n \ge 03$ there can be such pair of (c, n_0)

$F(n) = O(g(n)) = O(n)$ for $c = 9$, $n_0 = 1$

$F(n) = O(g(n)) = O(n)$ for $c = 7$, $n_0 = 3$

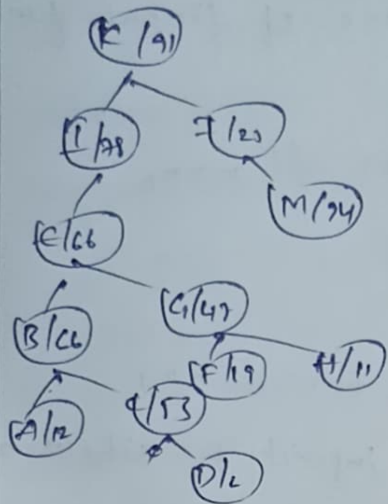
→ Find the lower bound of running time of Quadratic

$F(n) = 3n^2+2n+4$

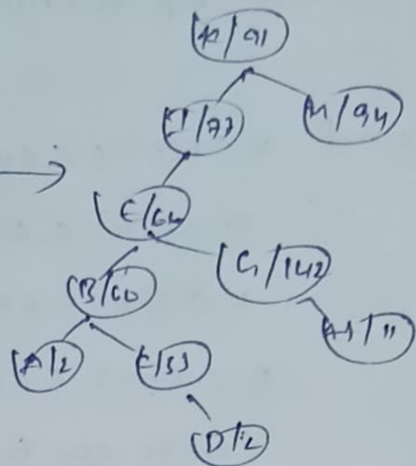
$0 \le c \times g(n) \le F(n)$

$0 \le c \times g(n) \le 3n^2+2n+4$

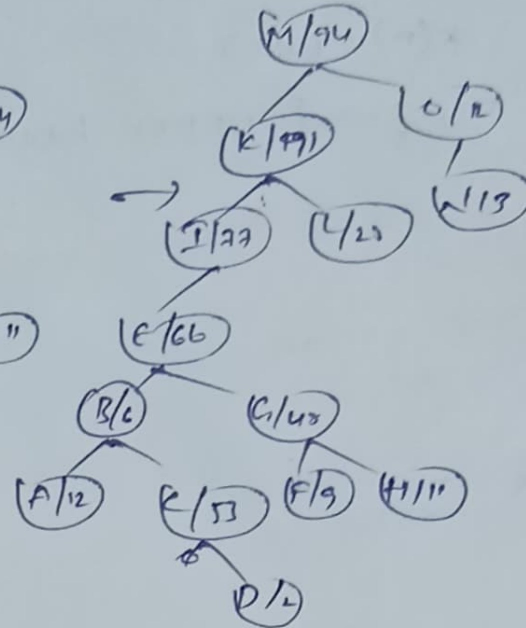
$0 \le 3n^2 \le 3n^2+2n+4$ → true $\forall n \ge 1$



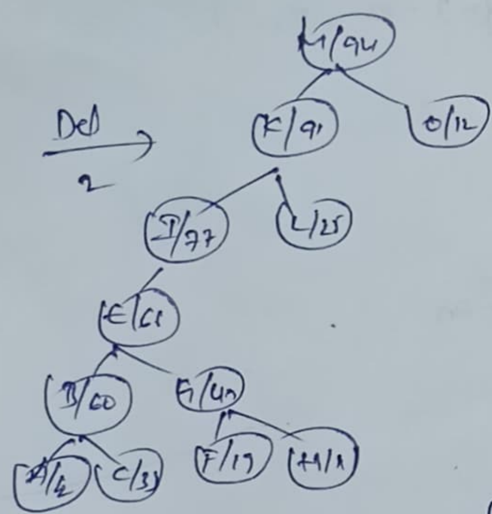
→



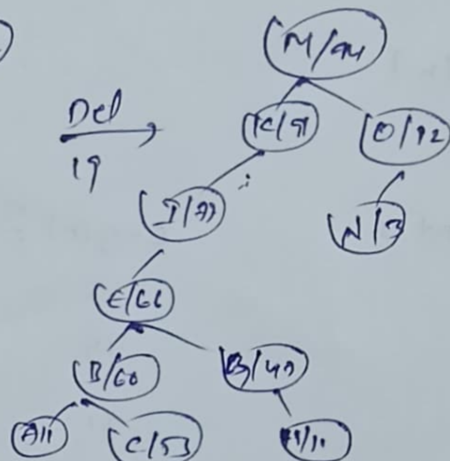
→



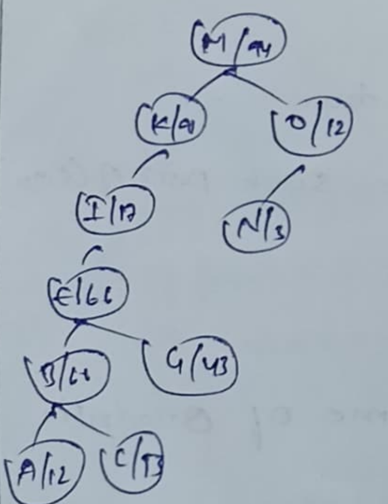
Del 2 →



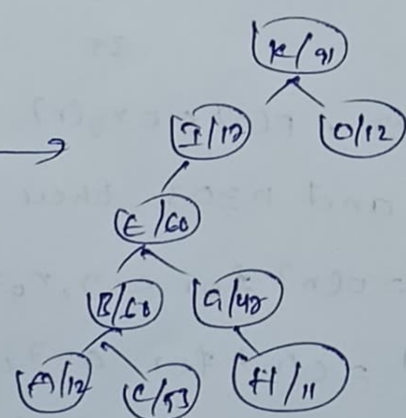
Del 19 →



Del 94 →



→



$0 \le cn^2 \le 3n^2+2n+4 \rightarrow$ true $\forall\ n \ge 1$

Above both inequalities are true and so such infinite inequalities

so, $F(n) = \Omega(g(n)) = \Omega(n^2)$ for $c=3$, $n_0=1$

$F(n) = \Omega(g(n)) = \Omega(n^2)$ for, $c=1$, $n_0=1$ and so on

Algorithm to find the max element in an array and its time complexity.

```
Algorithm max(arr,n,s,e,max){
    if (s==e){
        max=a[i];
    }
    elseif(e=s+1){
        if (a[s]<a[e]){
            max=a[e];}
        else{
            max=a[i];
        }
    }
else{
    mid=(s+e/2);
    maximum(s,mid,max);
    maxmin(mid+1,e,max2)
    if(ma1>max2){
        max=max1;
    }
    else{
    max=max2;
    }
```

Time complexity

$T(n) = 2T(n/2) + 2$

$T(n/2) = 2T(n/4) + 2$

sub $T(n) = 2[2T(n/4) + 2] + 2$

$= 4T(n/4) + 2(2) + 2 \quad = 2T(n/8) + 2$

sub. $4(2T(n/8) + 2) + 2$

$8T(n/8) + 4(2) + 2$

next $16T(n/16) + 2^4 + 2^3 + 2^2 + 2$

$2^i \; T(n/2^i) + 2^i + 2^{i-1} + \ldots 2 + 1 - 1$

$n/2^i = 1$

$n = 2^i \Rightarrow i = \log_2 n$

$2^{\log_2 n} T(1) + \underbrace{2^{\log_2 n} + 2^{\log_2 n - 1} + 2^{\log_2 n - 2} \ldots 2}$

$2^{\log_2 n} + 2^{\log_2 n + 1} - 2$

$2^{\log_2 n} + 2\,[2^{\log_2 n}] - 2$

$= n + 2[n] - 2$

$= 3n - 2$

$O(n)$ = for all 3 cases.