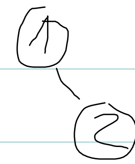
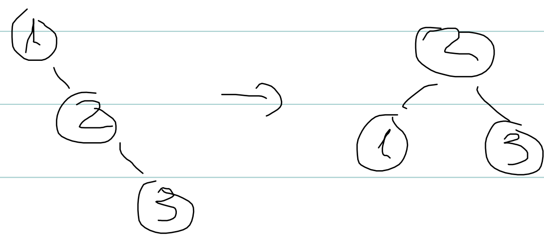
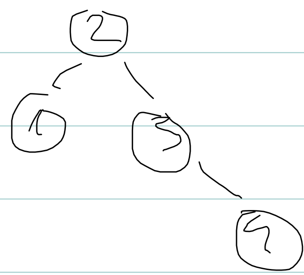
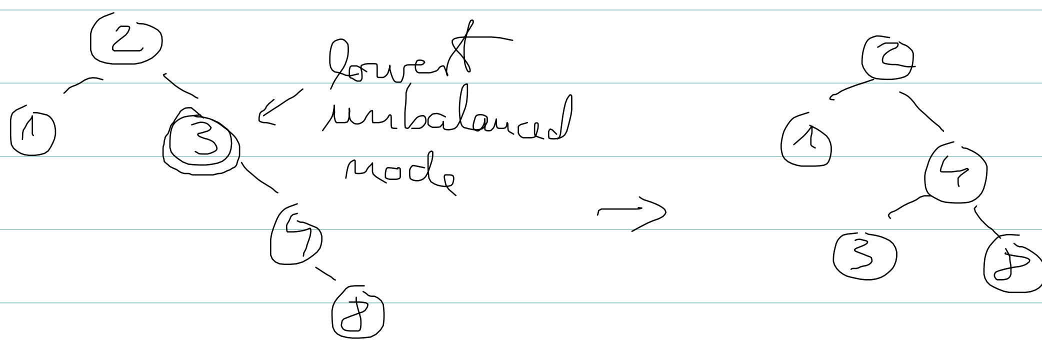
Q3: Insert 1 2 3 4 8 7 6 5 9,10 in an AVL tree:

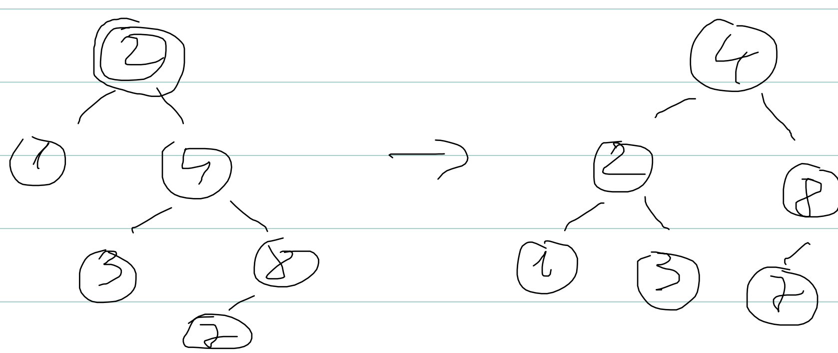
+1


+2


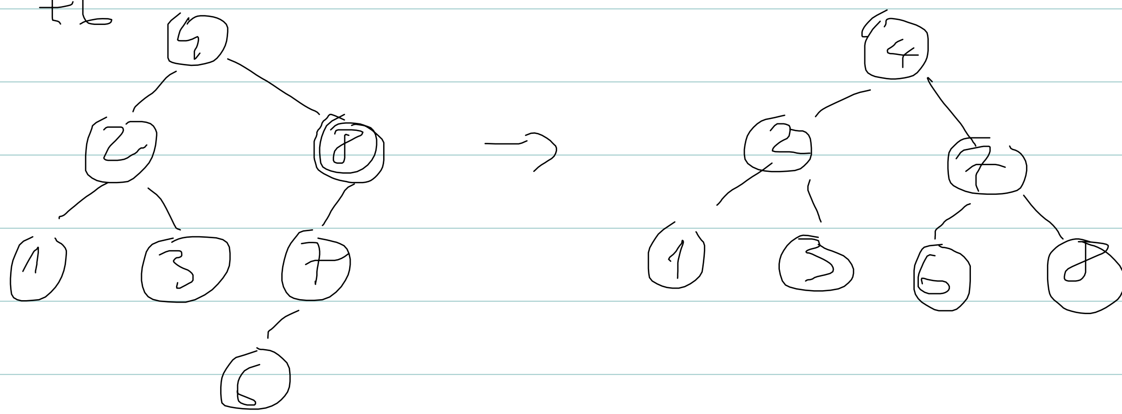
+3


+4


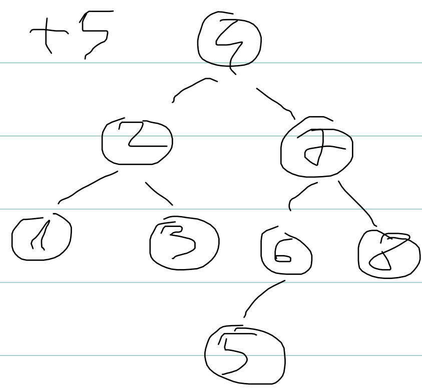
+8


+7


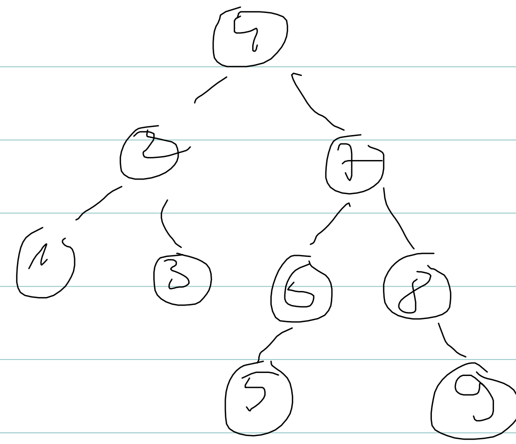
+6



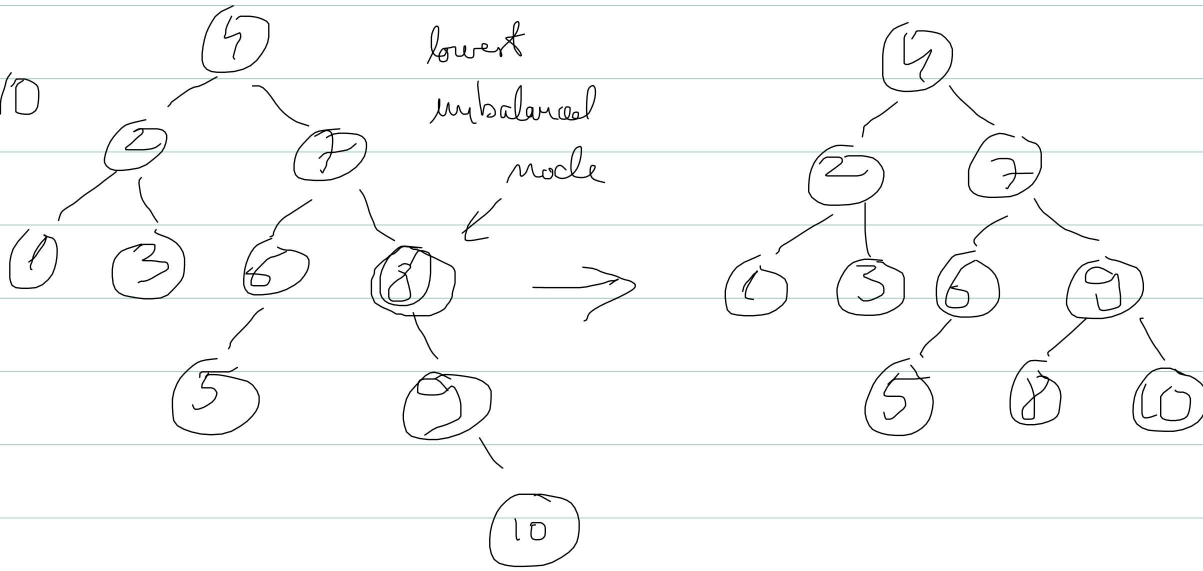
+5



+9

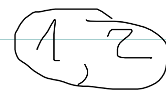


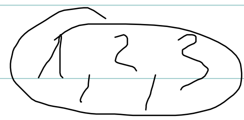
+10

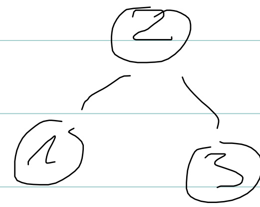


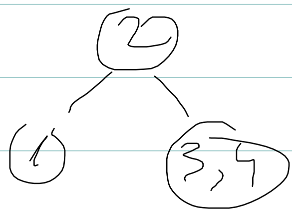
Q3.b) insert 1,2,3,4,8,7,6,5 9,10 in 2-3 tree:

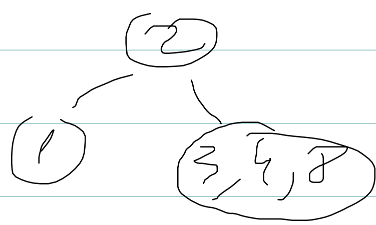
+1 

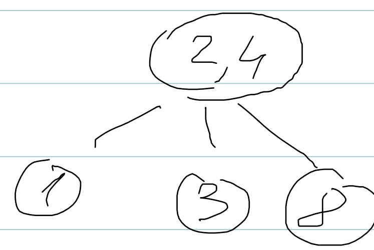
+2 

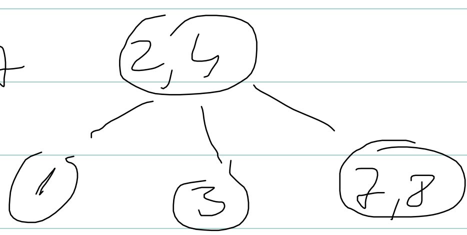
+3 

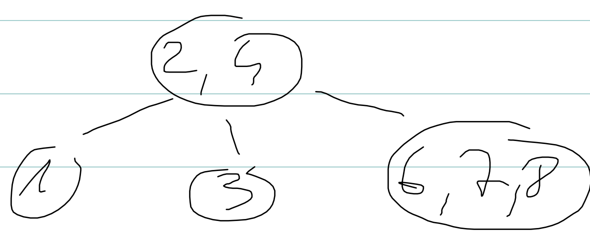


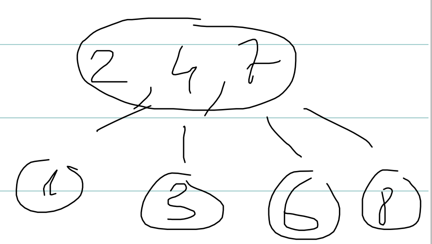
+4 

+8 

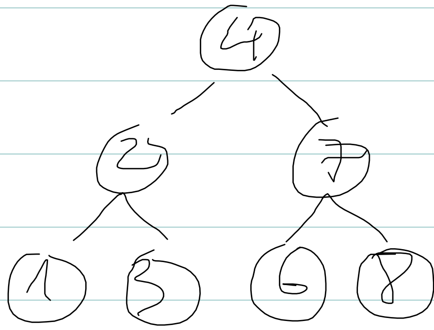


+7 

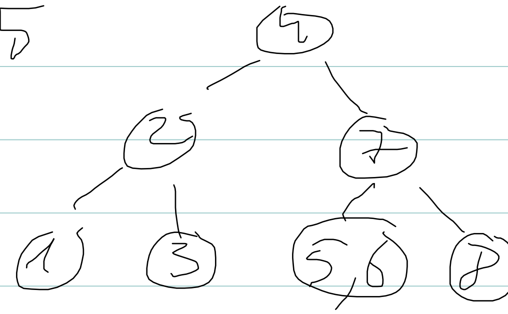
+6 



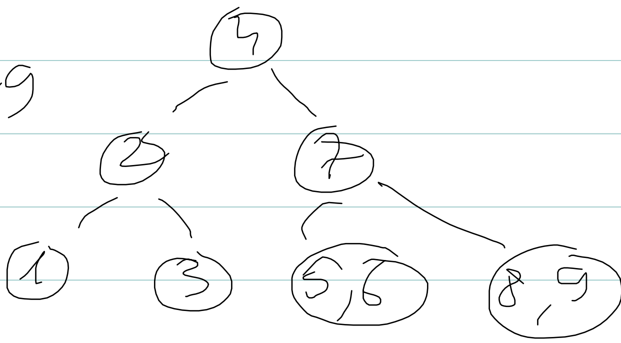
continued on next page



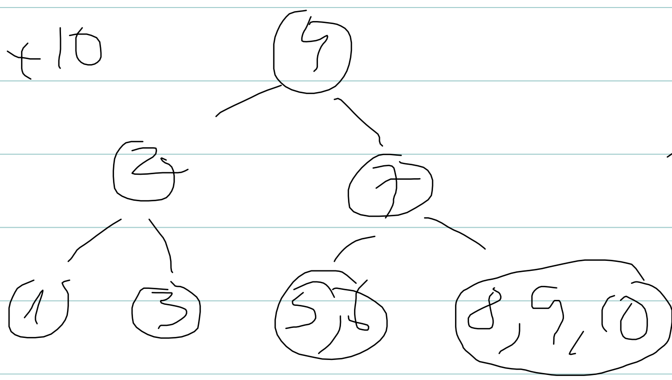
+5

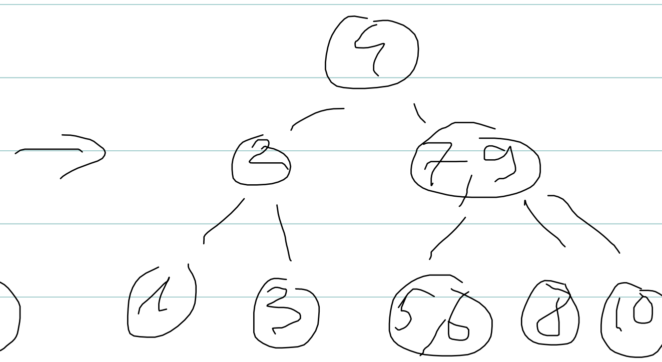


+9



+10





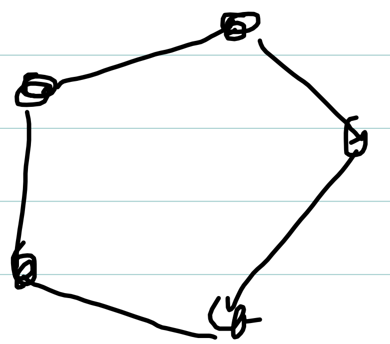
Marking scheme:
0.5 pt each correct insertion.

Total 5pts for each tree.

Q1. Bipartite graph check using DFS.

idea: to colour vertices with two colours, 0 & 1. No two adjacent vertices should share the same colour.

A graph is <u>not</u> bi-partite if there ∃ a cycle with odd length. Example:



– this is not a bipartite graph (cycle length 5)

The algorithm uses DFS in usual way. In addition, it assigns every visited vertex an alternating colour 0/1. & checks that back edges connect to vertices of opposite colour. Otherwise, an odd length cycle is present & graph is not bipartite.

Modifications for DFS algorithm:

- $\pi(v) = \begin{cases} 0 \text{ or } 1 & : \text{the vertex colour} \\ -1 & : \text{uncoloured} \end{cases}$
- no 'count' global variable. Current colour stored on stack

```
DFS(G):
    Π = -1   // all vertices are unvisited.
    forall v∈V do
        if Π(v) = -1 then dfs(v, 0)

dfs(v, c)      c → colour
    Π(v) ← c
    forall u ∈ N(v) do
        if Π(u) = -1 then dfs(u, (c+1) mod 2)
        else if Π(u) = c then
            throw exception "G not bipartite"
```

Marking scheme

[1 pt]: explanations
[2 pts]: correctly determining graph as bi-partite
[2 pts]: correctly determining a graph is NOT bipartite

Q2

It is the easiest to write the pseudocode recursively:

Powerset (S)
    if $|S| = 1$ then
        return $\{\emptyset, S\}$
    else
        Let $S' \subseteq S$ s.t. $S = \{s\} \cup S'$.
        $T \leftarrow$ Powerset (S')
        $T' \leftarrow \emptyset$
        forall $t \in T$ do
            $T' \leftarrow T' \cup (t \cup \{s\})$

        return $T \cup T'$

add s to all subsets in set T (store these sets in T')

Marking scheme

[1 pt] : clearly defined input & output

[2 pt] : correctly solving the base case (either $S = \emptyset$ or $|S| = 1$)

[2 pts]: correctly using the answer to the smaller instance