Assignment – 5

Answer to the Question No – 1

```
findMedian(A, B, la, ra, lb, rb):
{
        n = A.length;                   //As A and B have same length
        medA = floor( (la+ra)/2);
        medB = floor( (lb+rb)/2);
        if ( n == 1) return (A[0] + B[0])/2;
        else if (n == 2) return (max(A[0], B[0]) + min(A[1], B[1]))/2;
        else if (A[medA] > B[medB]) return findMedian(A, B, la, medA-1, medB+1, rb);
        else if (A[medA] < B[medB]) return findMedian(A, B, medA+1, ra, lb, medB-1);
        else return A[medA];
}
```

Answer to the Question No – 2

In a red-black tree, every path from the root to a leaf has the same number of black nodes, known as the black-height of the tree. Suppose, b is the black height of the red-black tree and h is the height of the red-black tree, then we can say, $b >= h/2$.

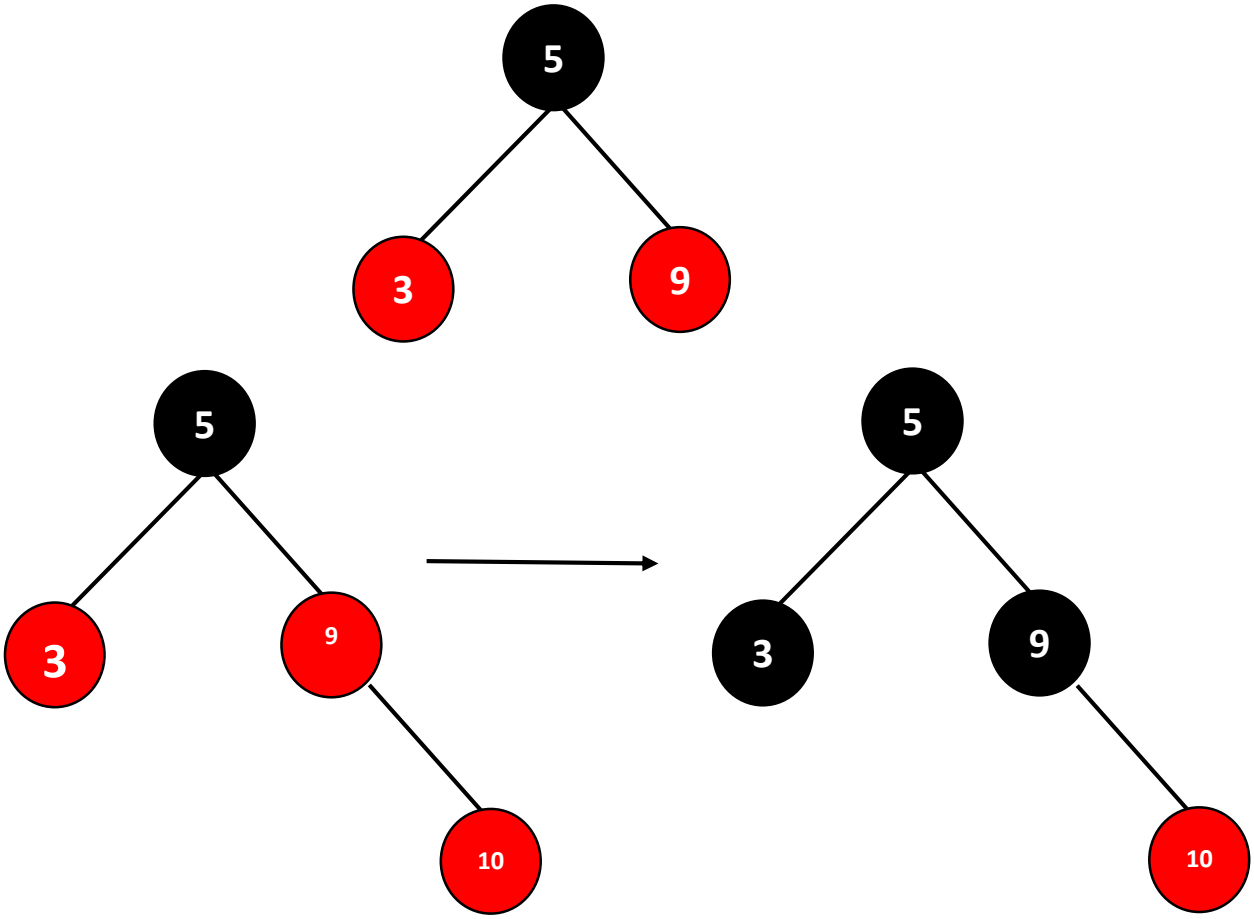
(a) To maximize the number of internal nodes, we want to minimize the height of the tree while still having a black-height of b. The shortest tree with a black-height of b will alternate between red and black nodes from the root to the leaves. This is because red nodes do not contribute to the black height, but they do increase the total height of the tree. for every black node, we can insert a red node as its child. So, the maximum height $h = 2b$ and this will have the maximum number of nodes for a black-height of b. Total number of internal nodes for this kind of tree is,

$$n = 2^h - 1 = 2^{2b} - 1.$$

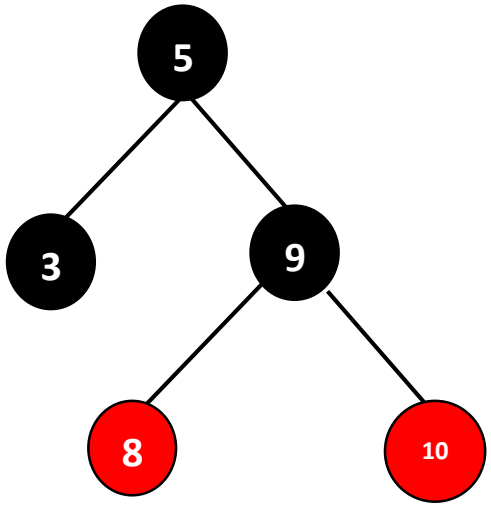
(b) To minimize the number of internal nodes, we want the tree to be as unbalanced as possible while maintaining the red-black tree properties. For such kind of tree, the height of the tree is minimized when it is a straight path, alternating between red and black nodes and the height of the tree will be b because each path to a leaf consists entirely of black nodes.
So, for minimum number of internal nodes the red-black tree will be a binary tree consisting of all black nodes and we have $b = h$ which gives total number of internal nodes = $2^h - 1 = 2^b - 1$.

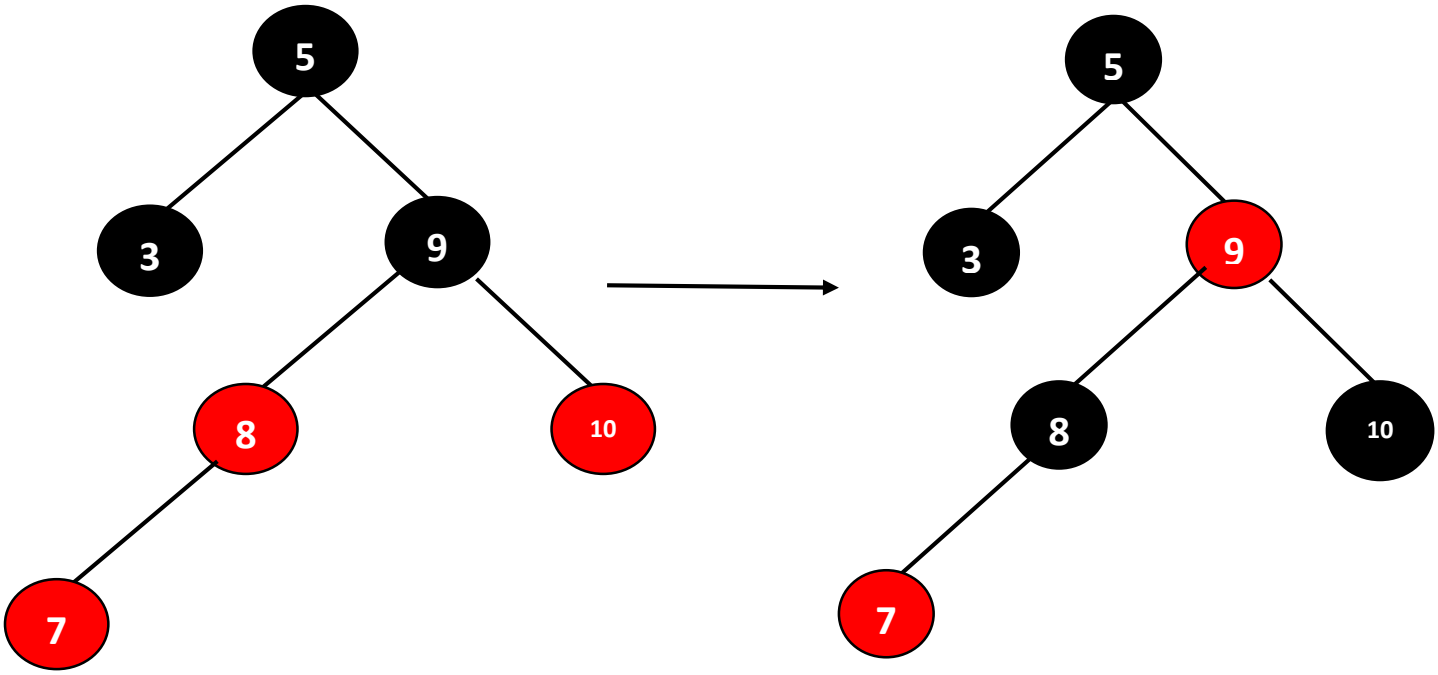
Answer to the Question No – 3

In the question, it is given that 5 is the root and it is black. Let's insert 9 and 3 as right and left child and both are red nodes. Since the root is black, there is no conflict. So, after 2nd iteration the tree looks like the following:

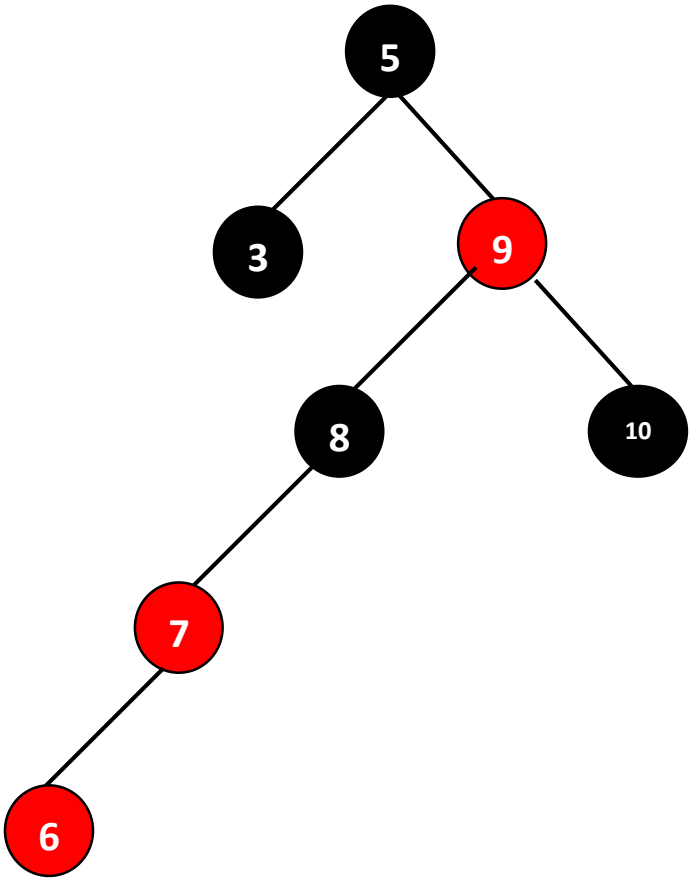


As 10 is greater than 5 and 9, it will be inserted in the right of 9 as red node. Since, two consecutive nodes are red, we need to recolor to maintain the red-black tree properties.





Again two consecutive red nodes (8 and 7) so recolor the parent and sibling of parent to get the proper red-black tree.



Here again 6 and 7 both are red nodes so this is the violation of red black tree. so we need to recolor the nodes. Before recoloring, we can see the sibling of 7 is null. So, at first we will rotate in right and then recolor the nodes. Doing this we will get the final tree as below: