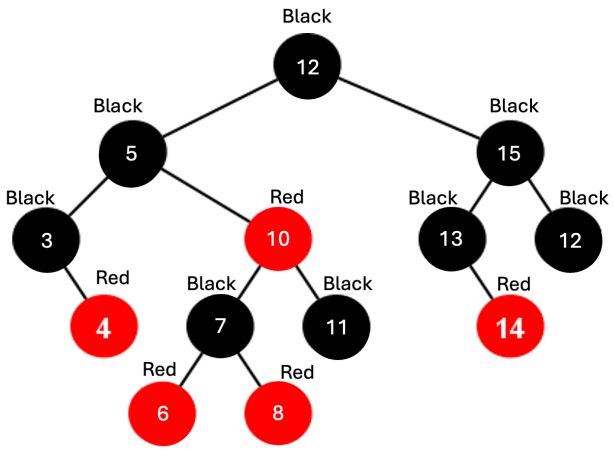
Quiz 3

Question 1: Choose the best ONE of these answers

(A) AVL Tree is a height balanced tree
(B) The maximum height of an AVL tree with N nodes is $\frac{\log(N)}{2}$
(C) The height difference between left and right children of any nodes in AVL tree is no more than 1.
(D) (A) and (B) are correct
(E) (A) and (C) are correct
(F) (B) and (C) are correct

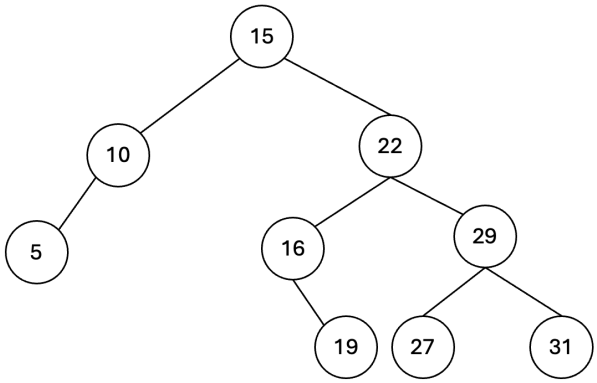
Question 2: Given a tree as follows:



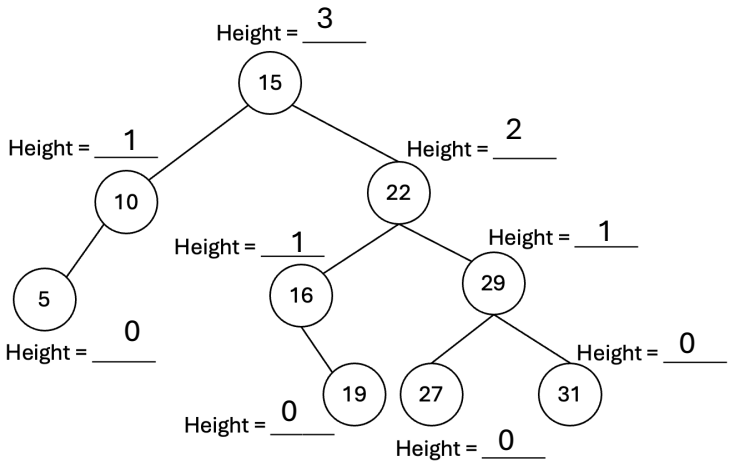
Is the above thee a red-black tree? Why?

Answer: No, it is not a red-black tree. The subtree 15, 13, 12 does not form a binary search tree property.

Question 3: Given the tree as follow?



Question 3.1: Calculate the height of each node

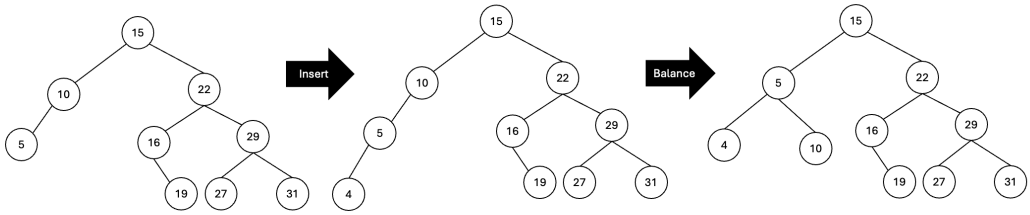


Question 3.2: Is the tree above an AVL tree? Why? If not, kindly perform the necessary adjustments to restore its AVL property?

Answer: Yes. The height difference between each node is no more than 1.

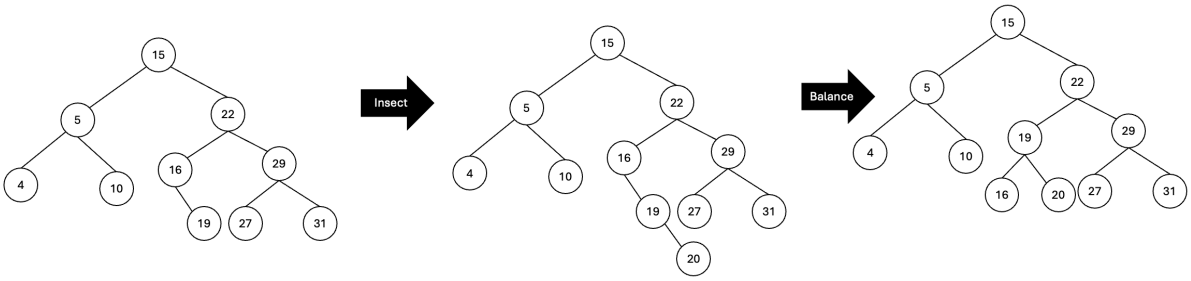
Question 3.3: Insert a value of 4 into the above AVL tree and balance it. If your answer in Question 3.2 is a no, please insert a value of 4 into your AVL Tree after your rebalancing.

Answer:



Question 3.4: Insert a value of 20 into the AVL obtained in Question 3.3 and balance it.

Answer:



Question 4: What are the operations that could be performed in $O(\log N)$ time complexity by red-black tree of N nodes?

(A) Insert, Search, Deconstructor

(B) Deconstructor, Delete, Search

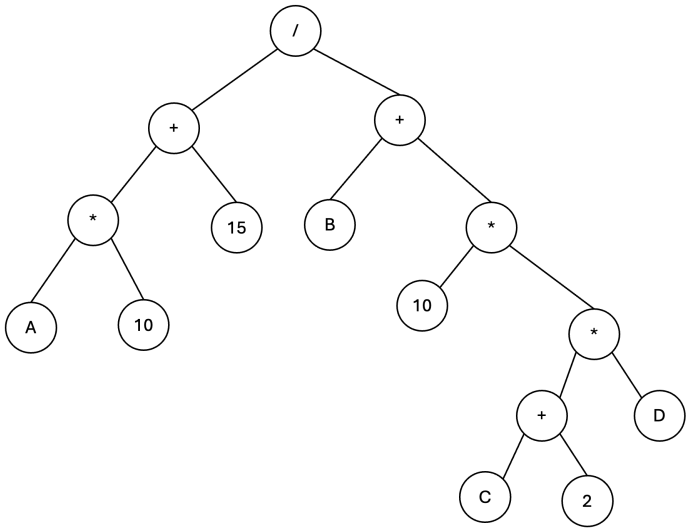
(C) Insert, Delete, Search

(D) Deconstructor, Insert, Delete

Question 5: Given the mathematical expression as follows $\frac{A*10+15}{B+\ 10(C+2)*D}$

Question 5.1: build the binary tree with binary operators to represent the above mathematical expression.

Answer:



Question 5.2: Choose the best ONE of these answers

(A) The leaf nodes are the operands and the non-leaf nodes are operators
(B) The leaf nodes are the operators and the non-leaf nodes are operands
(C) The leaf nodes and non-left nodes are all operands
(D) The leaf nodes and non-left nodes are all operators

Question 6: Given BSTNode and BST structures (below), write a function (pseudocode is acceptable) `bool BST::isAVL()` to check the current BST is whether a AVL tree or not.

```
struct BSTNode {
    int key;
    int height;
    BSTNode *left;
    BSTNode *right;
};

class BST {
    protected:
        BSTNode *root;
    public:
        BST();
        ~BST();
        bool isAVL();
};
```

Answer:

```
bool BST::isAVL() {
    std::queue<BSTNode*> queue;
    queue.push(this->root);
    while (!queue.empty()) {
        node = queue.front();
        queue.pop();
        if (node == NULL) continue;
        int leftHeight = node->left == NULL ? 0 : node->left->height;
        int rightHeight = node->right == NULL ? 0 : node->right->height;
        if (leftHeight - rightHeight > 1 || leftHeight -rightHeight < -1)
            return false;
        queue.push(node->left);
        queue.push(node->right);
    }
    return true;
}
```