ABHAYA.V
1BM18CS001

ADS Lab-8

Red black tree - insertion

Pseudo code:

```
enum  Color{RED, BLACK};
struct  Node
{
};

class  RBTree
{
    function calls;
};

void  RBTree::rotateLeft(Node *&root, Node *&pt)
{
    Node *pt_right = pt->right;
    pt->right = pt_right->left;
    if (pt->right != NULL)
        pt->right->parent = pt;
    pt->right->parent = pt->parent;
    if (pt->parent == NULL)
        root = pt_right;
    else if (pt == pt->parent->left)
        pt->parent->left = pt_right;
    else
        pt->parent->right = pt_right;
    pt->right->left = pt;
    pt->parent = pt->right;
}
```

ADS Lab 8

```
void RBTree::rotateRight(Node *&root, Node *&pt)
{
    Node *pt_left = pt->left;

    pt->left = pt_left->right;

    if(pt->left != NULL)
        pt->left->parent = pt;

    pt_left->parent = pt->parent;

    if (pt->parent == NULL)
        root = pt_left;

    else if (pt == pt->parent->left)
        pt->parent->left = pt_left;

    else
        pt->parent->right = pt_left;

    pt_left->right = pt;
    pt->parent = pt_left;
}

void RBTree::fixViolation(Node *&root, Node *&pt)
{
    Node *parent_pt = NULL;
    Node *grand_parent_pt = NULL;

    while((pt != root) && (pt->color != BLACK) &&
          (pt->parent->color == RED))
    {
```

```
parent_pt = pt -> parent;
grand_parent_pt = pt -> parent -> parent;
if (parent_pt == grand_parent_pt -> left)
{
    Node *uncle_pt = grand_parent_pt -> right;
    if (uncle_pt != NULL && uncle_pt -> color == RED)
    {
        grand_parent_pt -> color = RED;
        parent_pt -> color = BLACK;
        uncle_pt -> color = BLACK;
        pt = grand_parent_pt;
    }
    else
    {
        if (pt == parent_pt -> right)
        {
            rotate Left (root, parent_pt);
            pt = parent_pt;
            parent_pt = pt -> parent;
        }
        rotate Right (root, grand_parent_pt);
        swap (parent_pt -> color, grand_parent_pt -> color);
        pt = parent_pt;
    }
}
```

```
else
{
    Node * uncle_pt = grand_parent_pt->left;

    if((uncle_pt != NULL) && (uncle_pt->color == RED))
    {
        grand_parent_pt->color = RED;
        parent_pt->color = BLACK;
        uncle_pt->color = BLACK;
        pt = grand_parent_pt;
    }

    else
    {
        if (pt == parent_pt->left)
        {
            rotateRight(root, parent_pt);
            pt = parent_pt;
            parent_pt = pt->parent;
        }
        rotateLeft(root, grand_parent_pt);
        swap(parent_pt->color, grand_parent_pt->color);
        pt = parent_pt;
    }
}
root->color = BLACK;
}
```

ADS Lab 8

```
void BSTree::insert (const int& n)
{
    Node *pt = new Node(n);
    root = BSTinsert(root, pt);
    fixViolation(root, pt);
}
```