Kshitij R
1BM18CS04

ADS- Lab 7 (RB tree)

```
enum color {RED, BLACK};
struct node {
    int data;
    bool color;
    node *left, *right, *parent;
    node(int data) {
        this->data = data;
        left = NULL;
        right = NULL;
        parent = NULL;
        this->color = RED;
    }
};
class redblack {
    private:
        node *root;
    protected:
        void rotateleft(node *&, node *&);
        void rotateright(node *&, node *&);
        void fixviolation(node *&, node *&);
public:
    redblack {root = NULL;}
    void insertion(const int &n);
};
void redblack::rotateleft(node *&root, node *&parent)
    node *ptr = pt->right;
    pt->right = pt->right->left;
    if(pt->right != NULL)
        pt->right->parent = pt;
    ptr->parent = pt->parent;
```

Key

Kshitij R
1BM18CS048

```
if (pt→ parent == NULL)
    root = ptr;
else if(pt == pt→ parent→left)
    pt→parent→left = ptr;
    ptr→left = pt;
    pt→parent = ptr;
}
void redblack::rotateright(node *&root, node *pt) {
    node *ptr = pt→left;
    pt→left = pt→right;
    if (pt→ left != NULL)
        root = ptr;
    else if (pt == pt→parent→left)
        pt→parent→right = ptl;
    else
        pt→ parent→right = pt1;
    pt1 → right = pt;
    pt→ parent = ptl;
}
void redblack:: fixviolation (node*& root, node *&pt){
    node * parentpt = NULL;
    node * gparentpt = NULL;
    while ((pt != root) && (pt→ color != BLACK)
            && (pt→ parent→ color == RED)) {
        parentpt = pt→ parent;
        if (parentpt == gparent →left) {
            node * unclept = gparent pt→right;
            gparent→ color = RED;
            parentpt→color = BLACK;
            uncle pt→ color = BLACK;
            pt = gparent pt;
        }
```

Krishty R
1BM19CS048

```
else {
    if (pt == parentpt->right)
        rotateleft(root, parentpt);
        pt = parentpt;
        parentpt = pt->parent;
    }
    rotateright(root, gparentpt);
    swap(parentpt->color, gparentpt->color);
    pt = parentpt;
    }
}
else {
    node * unclept = gparentpt->left;
    if ((unclept != NULL) && (unclept->color == RED))
        gparentpt->color = RED;
        parentpt->color = BLACK;
        unclept->color = BLACK;
        pt = gparentpt;
    }
}
void * insertion (node *root, node *pt)
{ if (root == NULL)
        return pt;
    if (pt->data < root->data) {
        root->left = insertion(root->left, pt);
        root->left->parent = root;
    }
    else if (pt->data > root->data) {
        root->right = insertion(root->right, pt)
        root->right->parent = root;
    }
    return root;
}
```