Name: Anusree Manoj K
USN: 1BM18CS017
Date: 18-11-2020

LAB-8 – Program 6 – Insertion operation on Red Black Tree

```
Node* BSTInsert (Node *root, Node *pt)
{
    if (root == NULL)
        return pt;
    if (pt->data < root->data)
    {
        root->left = BSTInsert (root->left, pt);
        root->left->parent = root;
    }
    else if (pt->data > root->data)
    {
        root->right = BSTInsert (root->right, pt);
        root->right->parent = root;
    }
    return root;
}

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;
```


Anusree

```
    else
        pt->parent->right = pt_right;
    pt_right->left = pt;
    pt->parent = pt_right
}
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;
```


Anusree

Anusree Manoj K
IBM18CS017

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


Anusree

```
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;
}
```

Anusree Manoj K
IBM18CS017

```
void RBTree::insert(int &data)
{
    Node *pt = new Node(data);
    root = BSTInsert(root, pt)
    fixViolation(root, pt)
}
```


Anusree