Snehita. J
1BM18CS109

```
#include <bits/stdc++.h>
using namespace std;
enum Color {RED, BLACK};
struct Node
{   int data;
    bool color;
    Node *l, *r, *p;
    Node(int data)
    {   this->data = data;
        left = right = p = NULL;
        this->c = RED;
    }
};
void inorderHelper(Node *root
{   if (root == NULL)
        return;
    inorderHelper(r->left);
    cout << root->data << " ";
    inorderHelper(r->right);
}
Node * BSTInsert(Node *r, Node *pt)
{   if (!r) return pt;
    if (pt->data < r->data)
    {   r->left = BSTInsert(r->left, pt);
        r->left->parent = r;
    }
    else
        return root;
}
void level(Node *r)
{   if (!r) return;
    queue<Node*> q;
    q.push(root);
    while(!q.empty())
    {
```

Snehila. J
1BM18CS107

```
    Node *temp = q.front();
    cout << temp->data;
    q.pop();
    if( temp->left != NULL)
            q.push(temp->left);
    if ( temp->right != NULL)
            q.push(temp->right);
  }
}
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;
    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;
}
void RBTree :: fixViolation(Node *&root, Node *&pt)
{
    Node *p_pt = NULL;
    Node *g_p_pt = NULL;
    while (pt != root && pt->color != BLACK && pt->parent->color == RED)
    {
        p_pt = pt->parent;
        g_p_pt = pt->parent->parent;
```

Smehita. J
1BM18CS109

```
if (p_pt == g_p_pt -> left)
{   Node * uncle_pt = gp_pt -> right;
    if (uncle_pt != NULL)
    {
        g_p_pt -> colour = RED;
        p_pt -> color = BLACK;
        uncle_pt -> color = BLACK;
    }
    else {
        if (pt == p_pt -> right)
        {   rotateLeft (root, p_pt);
            pt = p_pt;
            p_pt = pt -> parent;
        }
        pt = p_pt;
    }
}
else {
    Node * u_pt = g_p_pt -> left;
    if ( u_pt != NULL && u_pt -> color == RED)
    {
        g_p_pt -> color = RED;
        p_pt -> color = BLACK;
        pt = gp_pt;
    }
    else {
        if ( pt == p_pt -> left )
        {   rotateRight (root, parent_pt);
            pt = p_pt;
            p_pt = pt -> parent;
        }
        pt = p_pt;
    }
    root -> color = BLACK;
}
```

Suchita J
IBM18CS109

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

int main()
{
    RBTree tree;
    tree.insert(7)
    tree.inorder();
    tree.level();
    return 0;
}
```