18/11/2020 ADS - LAB - 7 REVANTH-R
RED - BLACK - TREE 1BM18CS082

Insertion operation

```
class RBTree
{   private:    Node *root;
    protected:  void rotateLeft (Node *&, Node *&);
                void rotateRight (Node *&, Node *&);
                void fixViolation (Node *&, Node *&);
    public:
                RBTree () { root = NULL; }
                void insert (const int &n);
                void inorder ();
                void levelOrder ();
};

void inorder_Helper (Node *root)
{   if (root == NULL) return;
    inorder_Helper (root->left);
    cout << root->data << " ";
    inorder_Helper (root->right); }

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;
  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;
}
```

```
// Function to insert a new node with
   given data

void RBTree :: insert (const int &data)
{
    Node *pt = new Node (data);

    root = BSTInsert (root, pt);
    fixViolation (root, pt);
}

// Function to do inorder and level
   order traversals

void RBTree :: inorder ()
{ inorderHelper (root); }
void RBTree :: levelOrder ()
{ levelOrderHelper (root); }
```

Devtaneh