Akshay S Bharad..
1BM18CS011

ADS Lab-7 Writeup (Red-black trees)

```
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 inorder ();
              void levelorder();
 };
 void redblack :: rotateleft (node *&root, node *&pt)
      node * ptr = pt->right;
      pt -> right = pt->right -> left;
      if ( pt->right != NULL)
           pt -> right -> parent = pt;
      ptr -> parent = pt -> parent;
```

AS

Akshay S Bharadwaj
1BM18CS011

```
    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 *ptl = pt -> left;
    pt -> left = ptl -> right;
    if (pt -> left != NULL)
        pt -> left -> parent = pt;
    ptl -> parent = pt -> parent;
    if (pt -> parent == NULL)
        root = ptl;
    else if (pt == pt -> parent -> left)
        pt -> parent -> right = ptl;
    else
        pt -> parent -> right = ptl;
    ptl -> right = pt;
    pt -> parent = ptl;
}
void redblack :: fixviolation( node *&root, node *&pt){
    node * parentpt = NULL;
    node * gparent pt = NULL;
    while ((pt != root) && (pt -> color != BLACK)
            && (pt -> parent -> color == RED)) {
        parentpt = pt -> parent;
        gparent pt = pt -> parent -> parent;
        if (parentpt == gparentpt -> left) { //Case A
            node * unclept = gparentpt -> right;
            if (unclept != NULL &&   // Case 1
                unclept -> color == RED) {
```

[illegible]

Akshay S Bharadwaj
1BM18CS011

```
            gparentpt -> color = RED;
            parentpt -> color = BLACK;
            unclept -> color = BLACK;
            pt = gparentpt;
        }
        else {
            if (pt == parentpt -> right) { // Case 2
                rotateleft ( root, parentpt);
                pt = parentpt;
                parentpt = pt -> parent;
            } // Case 3
            rotateright (root, gparentpt);
            swap (parentpt -> color, gparentpt -> color);
            pt = parentpt;
        }
    } // Case B
    else {
        node *unclept = gparentpt -> left;
        if ((unclept != NULL) &&
            (unclept -> color == RED)) { // Case 1
            gparentpt -> color = RED;
            parentpt -> color = BLACK;
            unclept -> color = BLACK;
            pt = gparentpt;
        }
        else { // Case 2
            if (pt == parentpt -> left) {
                rotateright (root, parentpt);
                pt = parentpt;
                parentpt = pt -> parent;
            } // Case 3
            rotateleft (root, gparentpt);
            swap (parentpt -> color, gparentpt -> color);
            pt = parentpt;
} } }
```

AB

Akshay S Bharadwaj
1BM18CS011

```
    root -> color = BLACK;
}
void redblack :: insertion (const int &n) {
    node *pt = new node (n);
    root = insertion BST (root, pt);
    fixviolation (root, pt);
}
                          node *
node ** insertionBST (root, node *pt) {
    if (root == NULL) {
        return pt;
    if (pt -> data < root -> data) {
        root -> left = insertionBST (root -> left, pt);
        root -> left -> parent = root;
    }
    else if (pt -> data > root -> data) {
        root -> right = insertionBST (root -> right, pt);
        root -> right -> parent = root;
    }
    return root;
}
```

[illegible]