Batch 3

ADS LAB

1BM19CS403
CHIRANJEEVI

Red Block Tree

```
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  inorderHelper(Node *root)
  {
     if (root == NULL)
         return;
     inorderHelper(root->left);
     cout<< root->data<<" ";
     inorderHelper(root-> right);

   }
  Node *  BST Insert (Node* root, Node *pt)
        {    if (root==NULL)
                 return pt;
             if (pt->data < root->data)
             {
```

chf.

IBM19CS403
CHIRANJEEVI

```
        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   levelOrderHelper (Node *root)
  {
   if (root == NULL)
        return;
   std:: queue < Node *> q;
    q.push(root);
   while (!q.empty())
    {
      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 :: fixviolation ( Node * &root; Node * &pt)
   {
       Node * parent_pt = NULL;
```

②

chf.

1BM19CS403
CHIRANJEEVI

```
Node * grand-parent-pt = NULL;

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 -> colo == 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;

    }

  }
```

Chf.

IBM19CS403
CHIRANJEEVI

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

     }
```

Chf.