SHACINI-P
1BM19CS406.

Pseudocode for Insertion in a Red-Black tree

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

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

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

```
Node *pt_left = pt->left;
pt->left = pt_left->right;
if (pt->left != NULL)
    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;
}
```