Nilesh Gangopadhyay   1BM18CS058

Red black tree Insertion

```
void RBTree :: insert (const int& data)

{ Node * pt = new Node (data);
    root = BST Insert (root, pt);
    fix Violation (root, pt);
  }

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

         {   root -> left = BST Insert (root -> left          pt);
             root -> left -> parent = root;
         }

     else if (pt -> data > root -> data)
         {

             root -> right = BST Insert (root -> right, pt);
             root -> right -> parent = root;


         }
       return root;
     }
```

Case A : Parent of pt is left child of grand - parent of pt.

Case 1: The uncle of pt is also red only Recoloring required.

Case 2 : pt is right child of its parent left required

Case 3: pt is left child of its parent right - rotation required

Case B: parent of pt is right ~~[illegible]~~ right child of Grand parent of pt

Case 1: The uncle of pt is also red only recolouring required

Case 2 : pt is left child of its parent Right rotation required

Case 3: pt is right child of its parent then left rotation required