SNEHITA. I
IBM18CS109

```
Node *rightRot(Node *y)
{
    Node *x = y->left;
    Node *T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right))+1;
    x->height = max(height(x->left), height(x->right))+1;
    return x;
}

Node *leftRot(Node *x)
{
    Node *y = x->right;
    Node *T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(y->right))+1;
    return y;
}

int Balancefactor(Node *N)
{
    if (N == NULL) return 0;
    return height(N->left) - height(N->right);
}
```

SNEHITA · I
IBM18CS109

```
Node * insert (Node *node, int key)
{
    if(node == NULL)
        return newNode(key);
    if (key < node->key)
      node->left = insert(node->left, key);
    else if (key > node->key)
      node->right = insert(node->right, key);
    else
        return node;
    node->height = 1 + max( h(node->left), h(node->right));
    int b = BalanceFactor(node);
    if (b>1 && key < node->left->key) return rightRot(node);
    if (b<-1 && key > node->right->key) return leftRot(node);
    if (b>1 && key > node->left->key)
    {  node->left = leftRot(node->left;
       return rightRot(node);
    }
    return node;
}

DELETE( node *p, int data)
{ if (p->left == NULL && p->right == NULL)
   { if (p == this->root)
          this->root = NULL;
     delete p;
     return NULL;
   }
   if (p->data < data)
      p->right = delete(p->right, data);
   else if (p->data > data)   p->left = delete(p->left, data);
```

SNEHITA·I
IBM18CS109

```
else
{
    if(p->left != NULL)
    {
        q = inpre(p->left);
        p->data = q->data;
        p->left = delete(p->left, q->data);
    }
    else
    {
        q = insuc(p->right);
        p->data = q->data;
        p->right = delete(p->right, q->data);
    }
}
return p;
}
```