PROGRAM - 4

```
Node *rightRoot (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 *leftRoot (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);
}
```

```
insert (Node *node, int Key)
{
    if (node == NULL) return new Node(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 rightRoot(node);
    if (b < -1 && Key > node->right->Key)
        return leftRoot(node);
    if (b > 1 && Key > node->left->Key)
    {
        node->left = leftRoot(node->left);
        return rightRoot(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->
    p->right = delete(p->right, data);
else if (p->data > data)
    p->left = delete(p->left, data);
else
{
    if (p->left != NULL)
    {
        q = insert(p->left);
        p->data = q->data;
        p->left = delete(p->left, q->data);
    }
    else
    {
        q = insert(p->right);
        p->data = q->data;
        p->right = delete(p->right, q->data);
    }
}
return p;
}
```