Prog 5 : 2-3 tree.

```
Class tree
{
    TreeNode *root = NULL;
    Public :
        void traverse ()
        {
            if (root != NULL)
                root -> traverse ();
        }
        void insert (int k);
        void remove (int k);
    };
Tree Node :: TreeNode (bool leaf1)
{
    leaf = leaf1;
    keys = new int [3];
    Child = new Treenode*[4];
    n=0;
}
int TreeNode :: find key (int k)
{
    int idx =0;
    While (idx < n && keys [idx]< k)
        ++idx;
    return idx;
}
Void Tree :: Insert (int k)
{
    if(root == NULL)
    { root = new TreeNode (true);
        root -> keys [0] = k;
        root -> n = 1;
    }
```

```
else {
    if (root->n == 3)
    {
        TreeNode *s = new TreeNode (false);
        s->child[0] = root;
        s->splitchild (0, root);
        int i = 0;
        if (s->keys[0] < k)
            i++;
        s->child[i]->insertNonfull(k);
        root = s;
    }
    else
        root->InsertNonfull(k);
void TreeNode :: InsertNonfull(int k)
{
    int i = n-1;
    if (leaf == true)
    {
        while (i>=0 && keys[i] > k)
        {
            keys[i+1] = keys[i];
            i--;
        }
        keys[i+1] = k;
        n = n+1;
}
else {
    while (i>=0 && keys[i]>k)
        i--;
    if (child[i+1]->n == 3)
    {
        splitchild (i+1, child[i+1]);
        if (keys[i+1] < k)
            i++;
    }
    child[i+1]->insertNonfull(k);
```

```
void TreeNode :: remove (int k)
{
    int idx = findKey (k);
    if (idx < n && keys [idx] == k)
    {
        if (leaf)
            removeFromLeaf (idx);
        else
            removeFromNonLeaf (idx);
    }
    else
    {
        if (leaf)
        {
            cout << "The key doesn't exist" << endl;
            return;
        }
        bool flag = ((idx == n) ? true : false);
        if child [idx] → n < 2)
            fill (idx);
        if (flag && idx > n)
            child [idx - 1] → remove(k);
        else
            child [idx] → remove (k);
    }
    return;
}
void TreeNode ::removeFromLeaf (int idx)
{
    int k = keys [idx];
    if (child [idx] → n >= 2)
    {
        int pred = getPred (idx);
        keys [idx] = pred;
        child [idx] → remove (pred);
    }
```

```
else if (child[idx+1]->n >= 2)
{
    int succ = getSucc(idx);
    key[idx] = succ;
    child[idx+1]->remove(succ);
}
else {
    merge(idx);
    child[idx]->remove(k);
}
    return;
}
void TreeNode :: removeFromLeaf (int idx)
{
    for (int i = idx+1; i<n; ++i)
        keys[i-1] = keys[i];
    n--;
    return;
}
```