Name - Nimit Sogal
USN - 1BM18CS063
Date - 3/11/20.

ADS - Lab - (5) - 2-3 Trees

```
class TwoThreeTree {
    TwoThreeNode *root;
    int t;
    TwoThreeTree(){
        root = NULL;
        t = 2;
    }
}
void insert (int k){
    if (!root){
        root -> keys[0] = k;
        root -> n = 1;
    }
    else if (root -> n == 2*t - 1){
        TwoThreeNode *s = new TwoThreeNode()
        s -> c[0] = root;
        s -> split (0, root);
        int i = 0;
        if (s -> keys[0] < k)
            i++;
        s -> c[i] -> insertNode (k);
        root = s;
    } else {
        root = insertNode (k);
    }
}
```

Name - Nimit Sejpal.
USN - 1BM18 CS063.

```
void insertNode(int k){
    int i = n-1;
    if ( ) {
        while (i>=0 && keys[i] > k){
        keys[i+1] = keys[i];
        i--;
    }
    else {
        while (i>=0 && keys[i] > k)
            i--;
        if ( (i+1) -> n == n*t -1)
            split (i+1, c[i+1]).
        if (keys[i+1] < k)
            i++;
    }
    c[i+1] -> insertNode(k);
    }
}

void split (int i, TwoThreeNode *y){
    TwoThreeNode *z = new TwoThreeNode(y->t, y->leaf);
    z -> n = t-1
    for (int j = 0; j < t-1; j++)
        z -> keys[j] = y -> keys[j+t];
    if (y -> leaf == false)
    {   for (int j = 0; j<t; j++)
            z -> c[j] = y -> c[j+t];
    }
```

Name - Nimal Sajal
USN - 1BM18CS063.

```
y->n = x->n-1;
for (int j = n; j >= i+1; j--)
    c[j+1] = c[j];
c[i+1] = z;
for (int j = n-1; j >= i; j--)
    keys[j+1] = keys[j];
keys[i] = y->keys[x->n-1];
n = n+1;
}
void remove(int x){
if (!root)
{
    cout << "Tree empty";
    return;
}
root->remove(x);
if (root->n == 0) {
    TwoThreeNode *temp = root;
    if (root->leaf)
        root = NULL;
    else
        root = root->c[0];
    delete temp;
}
return;
}
```

USN - 1VM18CS061.

```
void removeFromLeaf (int idx){
    for(int i=idx+1; i<n; i++){
        keys[i-1] = keys[i];
    n--;
    }
    return;
void removeFromNonLeaf (int idx){
    int k = keys[idx];
    if (C[idx]->n >= t){
        int pred = getPred(idx);
        keys[idx] = pred;
        C[idx] -> remove(pred);
    }
    else if(C[idx+1]->n >= t){
        int succ = getSucc(idx);
        keys[idx] = succ;
        C[idx+1] -> remove(succ);
    }
    else{
        merge(idx);
        C(idx) -> remove(k);
    } return;
    }
```