Sanjeev Kumar Singh
1BM18CS093
B5

28/10/20

Lab 6

2-3 tree

```
Void Tree: insert (int K)
{ if (root == NULL) {
   root = new TreeNode (tree);
   root → Keys[0] = K;
   root → n = 1; }
else {
   if (root → n == 3) {
   Treenode *S = new Tree node (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 → insert nonfull (K); }
}
Void Tree node :: insertnonfull (int K){
int i = n-1;
if (leaf == = true) {
while (i >= 0 && Keys [i] > K) {
   Key[i+1] = Key [i];
      i--;
   }
   Key [i+1] = K;
   n = n+1; }
```

```
else {
if ( i>=0 && key[i] >k)
    i--;
    if (child[i+1] -> n==3) {
splitchild (i+1, child[i+1]);
    else (key[i+1] <k);
    i++;
}
child[i+1] -> insertnonfull (k);
}
}
```

Delete

```
void Tree node :: remove (int k) {
    int idx = find_key (k)
    if (idx < n && key[idx] == k) {
    if (leaf) remove from leaf (idx);
    else remove from non-leaf (idx);
    }
else if (leaf) {
Cout << " keys doesn't exist << endl;
    return;
}
bool flag = ((idx == n ? true : false);
if (child[idx] -> n < 2)
    fill (idx); fills child [idx];
if (flag && idx > n)
    child [idx-1] -> remove [k];
else
    child[idx] -> remove [k]; }
return;
}
```