B true insertion

1BM18CS016
Ankitha

```
void BTree::insert(int k)
{
    if (root == NULL)
    {
        root = new BTreeNode(t, true);
        root->key[0] = k;
        root->n = 1;
    }
    else
    {
        if (root->n == 2*t - 1)
        {
            BTreeNode *s = new BTreeNode(t, false);
            s->C[0] = root;
            s->splitChild(0, root);
            int i = 0;
            if (s->key[0] < k)
                i++;
            s->C[i]->insertNonFull(k);
            root = s;
        }
        else
            root->insertNonFull(k);
}
void BTreeNode::insertNonFull(int k)
{
    int i = n-1;
    if (leaf == true)
    {
        while (i >= 0 && keys[i] > k)
        {
            key[i+1] = keys[i]
            i--;
        }
        key[i+1] = k;
        n = n+1;
    }
```

```
else
{
    while (i>=0 && key[i]>k)
    {   i--;   }
    if (C[i+1]->n == 2*t-1)
    {
        Split Child (i+1, C[i+1]);
        if (keys[i+1]<k)
            i++;
    }
    C[i+1]->insertNonFull(k);
}
}
void BTreeNode::SplitChild(int i, BTreeNode *y)
{
    BTreeNode *z = new BTreeNode(y->t, y->l
    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+1];
    }
    y->n = t-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--)
        key[j+1] = keys[j];
```

```
        key[i] = y->keys[t-1];
        n = n+1;
    }

class BTreeNode
{
    int *keys;
    int t;
    BTreeNode **C;
    int n;
    bool leaf;
public:
    BTreeNode(int _t, bool leaf);
    void insertNonFull(int k)
    void splitChild(int i, BTreeNode *y);
};
```