MD YASEEN AHMED
1BM19CS404.

✶ <u>BTree Insertion:</u>
→ BTreeNode

```
        int n, t, *keys;
        BTreeNode **c;
        bool leaf;


    BTree


        BTreeNode *root;
        int t;

    BTreeNode :: BTreeNode (int t1, bool l)
    {
        t = t1;
        leaf = l;

        keys = new int [2*t - 1];
        c = new BTreeNode * [2*t];
        n = 0
    }
```

```cpp
void BTree :: insert (int k)
{
    if (root == NULL)
    {
        root = new BTreeNode (t, true);
        root -> Keys[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;
        }
```

```cpp
    else {
        root->insertNonFull(k);
    }
}
}

void BTreeNode:: insertNotFull (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 (C[i+1] ->n == 2*t-1)
{
    splitChild (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->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];
    }
}
```

```
y -> n = t - 1;

for (int j = n; j >= i + 1; j--)
    c[j+1] = c[j].

c[j+1] = z;

for (int j = n - 1; j >= i; j--)
    Keys[j+1] = Keys[j];

Keys[i] = y -> Keys[t-1];

n = n + 1;
}
```