VYSHNAVI.M.N
1BM18CS125

```
class BTreeNode
{
    int *keys;
    int t, leaf;
  public:
    BTreeNode(int t, bool leaf);
    void insertNonFull(int k);
    void splitChild(int i, BTreeNode *y);
    void traverse();
    BTreeNode *search(int k);
    friend class BTree;
};

class BTree
{
    BTreeNode *r;
    int t;
  public:
    BTree(int t)
    {
        root = NULL;
    }
    void traverse()
    {
        if (root != NULL)
        root->traverse();
    }
    void insert(int k);
};
```

```
BTreeNode :: BTreeNode (int t1, bool leaf1)
{
    t = t1;
    leaf = leaf1;
    keys = new int [2 * t];
    C = new BTreeNode * 2 [2 * t];
    n = 0;
}
void BTreeNode :: traverse ()
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (leaf == false)
            C[i] -> traverse();
            cout << " " << keys[i];
    }
    if (leaf == false)
        C[i] -> traverse();
}

BTreeNode * BTreeNode :: search (int k)
{
    int i = 0;
    while (i < n && k > keys[i])
        i++;
    if (keys[i] == k)
        return this;
    if (leaf == true) return NULL;
    return C[i] -> search(k);
}
```

```
BTreeNode : insert (int k)
{
    if (!root)
    {
        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 -> keys[0] < k) i++;
            s -> c[i] -> insertNonFull (i);
            root = s;
        }
    else root -> insertNonFull (k);
}
```