Program-06

Implementation of B-tree

```
class BTree
{
  BTreeNode *root;
  int t;
public:
    BTree(int _t)
    { root = NULL; t = _t; }

    void traverse()
    { if (root != NULL)
        root->traverse(); }

    BTreeNode* search(int k)
    { return (root == NULL)? NULL: root->search(k); }

    void insert(int k);
};

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->keys[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)
        {
            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[i+1] = z;
```

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

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

n = n+1;
}
```