```
class BTree {
    BNode *root;
    int t;
    BTree() {
        root = NULL;
        t = 2;
    }
}

void insert(int k) {
    if (!root) {
        root = new BNode();
        root->keys[0] = k;
        root->n = 1;
    } else if (root->n == 2*t - 1) {
        BNode *S = new BNode();
        S->C[0] = root;
        S->splitChild(0, root);
        int i = 0;
        if (S->keys[0] < k)
            i++;
        S->C[i]->insertIntoNode(k);
        root = S;
    } else {
        root = insertIntoNode(k);
    }
}
```

```
void insertintoNode(int k) {
  int i = n-1;
  if (is leaf) {
    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)
      split (i+1, c[i+1])
    if (keys[i+1] < k)
      i++;
  }
  c[i+1] -> insertintoNode(k);
}
}

void splitCint i,  BTree * y) {
  BTree *z = new BTree (y -> key);
  z -> n = t-1;
  for (int j=0; j<t-1; j+t)
    z -> keys[i] = y -> keys[j+t];
  if (y -> leaf == false)
  {
    for (int j=0; j<t; j++)
      z -> c[j] = y -> [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;
}
```