B-trees

Nidhish Lakhinana
1BM18CS059

insertion,

```
void insert (int k) {
        if (root == NULL) {
            root = new Node (t, true);
            root -> keys [0] = k;
            root -> n = 1;
        } else {
            if (root -> n == 2*t - 1){
              Node *s = new Node (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 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) {
            split child (i+1, c[i+1]) ;

            if (keys [i+1] < k)
                i++ ;

        }
        c [i+1] -> insert Non full (k) ;
    }
}

void   splitchild (i, Node *y) {
    Node *z = new Node (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;
}
```