Lab 7 - Write a program to implement insertion operation on a B-tree

1) Intialize x as root
2) While x is not leaf do following
    a) Find the child of x that is going to be traversed next. Let child be y.
    b) If y is not full, change x to point y
    c) If y is full, split it & change x to point to one of two part of y. If k is smaller than mid key in y then set x as the first part of y. Else second part of y. When we split y we move a key from y to its parent x.
3) The loop in step 2 stops when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance. So simply insert k to x.

Pseudocode

```
 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);
  }
}
insertNonfull (int k)
{
   int i=n-1;
   if (leaf == true)
   {
      while (i>=0 && keys[i]>k]
      {
        keys[i+1] = keys[i];
          i--;
      }
    key[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 splitchild(int i, BTreeNode *y)
 {
    BtreeNode *z=new BTreeNode(y->t, y->leaf);
    z->n=t-1;
   for(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 (j=n; j>=i+1; j--)
    c[j+1] = c[j];

c[i+1] = z;

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

keys[i] = y ⇔ keys[t-1];
n=n+1;
}
```