Prog 6: Prog to impliment insertion operⁿ on a B-Tree

```
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 -> insert NonFull (k);
       }
    }
void BTreeNode :: insert NonFull (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 = newBTreeNode (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
}.
```