Snehita. I
IBM18CS109

```
class BTreeNode
{
    int *keys;
    int t in, leaf;
  public:
    BTreeNode(int_t, bool_leaf);
    void insertNonFull(int k);
    void splitchild(int i, BTreeNode *y);
    void traverse();
    BTreeNode *search(int k);
  friend class BTree;
};
class BTree
{
    BTreeNode *r;
    int t;
  public:
    BTree(int_t)
    { root = NULL; t = _t;}
    void traverse()
    {    if(root != NULL) root → traverse(); }
    void insert(int k);
};
BTreeNode :: BTreeNode (int t1, bool leaf1)
{
      t = t1;
     leaf = leaf1;
     keys = new int[2*t-1];
      c = new BTreeNode * 2[2*t];
      n=0;
}

void BTreeNode :: traverse()
{      int i;
```

Snehita. I
1BM18CS109

```
  for(i=0; i<n; i++)
  {    if (leaf ==false)
           c[i]→traverse();
       cout<< "   "<< keys[i];
   }
   if (leaf == false)
        c[i]→ traverse();
  }

BTreeNode  * BTreeNode :: search(int k)
{
       int  i=0;
       while(i<n && k > keys[i])
              i++;
       if ( keys[i]==k)
              return this;
       if (leaf == true)  return NULL;
       return  c[i]→ search(k);
 }
void  BTree::insert (int k)
{
     if ( !root)
      {    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)
```

Snehila . I
1BM18CS109

```
void BTreeNode :: insertNonFull (int k)
{
        int  i=n-1;
        if (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--; }
                  splitChild (i+1, c[i+1]);
                  if (keys[i+1]<k)   i++;
                  c[i+1] -> insert NonFull(k);
        }
}
void  BTreeNode :: 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];
      y->n=t-1;
      for( j=n ; j>= i+1; j--)
          c[j+1] = c[j];
      c[i+1]=z;

      n = n+1;
}
int  main()
{     BTree t(3);
      t.insert(10)
          :
       t.traverse();
  }
```