LAB 5

Nikhil.A.S
USN:1BM18CS061
2nd Nov 2020.

```
class TreeNode
{
    int *keys;
    TreeNode **child;
    int n;
    bool leaf;
    friend class Tree;
};
class Tree
{
    TreeNode * root = Null;
    public:
        void traversal(){
            if (root != Null)
                root->traversal();
        }
        void insert(int k);
        void remove(int k);
}
void Tree insert (int k)
{
    if (root == Null)
    { root = new TreeNode(true);
      root->key[0] = k
      root->n = 1;
    }
```

```
else {
  if (root->n == 3)
  {
    TreeNode *s = newTreeNode(false);
    s->child[0] = root.
    s->splitchild(0, root);
    int i=0
    if (s->key[0] < k)
        i++;
    s->child[i]-> insertNonFull(k);
    root = s;
  }
  else
  {
    root-> insertNonFull(k).
  }
}
}
Void TreeNode :insertNonFull(int k)
{ int i=n-1;
  if (leaf == true)
  { while (i>=0 && keys[i]>k)
    { keys[i+1] = key[i];
      i++;
    }
    key[i+1] =k;
    n=n+1;
}
```

```
else {
  while (i >= 0 && keys[i] > k)
    i--;
  if (child[i+1]->n == 3)
  { splitchild(i+1, child[i+1]);
    if (keys[i+1] < k)
      i++;
  }
  child[i+1]->insertNonFull(k);
}
}
void TreeNode::splitchild(int i, TreeNode key)
{ TreeNode *z = new TreeNode(y->leaf)
  z->n = 1;
  z->keys[0] = y->keys[2];
  if (y->leaf == false){
  for(int j = n; j >= i+1; j--)
    child[j+1] = child[j];
  child[i+1] = z;
  for(int j = n-1; j >= i; j--)
    keys[j+1] = keys[j];
  keys[i] = y->keys[1];
  n = n+1;
}.
```

```
void Tree :: remove (int k)
{
  if (! root)
  {
    cout << "Tree is empty" << endl;
    return;
  }
  root -> remove (k);
  if (root -> n == 0)
  {
    TreeNode *temp = root;
    if (root -> leaf) root = Null;
    else root = root -> child[0];
    delete temp;
  }
  return;
}
```