Write-Up

Guru Nanma
1BM18CS031

```
class TreeNode {
      int *keys;
      TreeNode **child;
       int n;
       bool leaf;
       // function declarations
       friend class Tree;
}
class Tree {
    TreeNode *root = NULL;
    public :
           void traverse()
                  if root != NULL
                        root -> traverse()
           void insert (int)
           void remove(int)
}
void Tree : insert (int k)
       if root == NULL
            root = new TreeNode (true)
            root -> keys [0] = k
            root -> n = 1
       else
          if (root -> n == 3)
               TreeNode *n = new TreeNode (false)
               n -> child[0] = root
               n -> splitchild (0, root)
               int i=0
```

Gaurav Mamta

```
        if ( n-> keys[0]  < k)
              i++
          n->child[i] -> insertNonFull(k)
          root = n
        else
            root -> insertNonFull(k)

void   TreeNode :: insertNonFull(k)
       int  i=n-1
       if  leaf = =true
           while (i >=0  and  keys[i]>k)
                   keys[i+1] = keys[i]
                   i--
           keys[i+1] = k
            n += 1
       else
           while (i>=0  and 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 *y )

         TreeNode *z  = new TreeNode (y -> leaf)
           z -> n = 1;
           z -> keys[0] = y -> keys[2]
```

GURU NANMA

```
if(y → leaf==false)
        for (int j = 0 < 2)
            z → child [j] = y→child [j + 2]

 y → n = 1
 for ( j = n to i+1 ; j --)
        child [j+1] = child [j]
 child [i+1] = z
 for ( j = n-1 to i ; j--)
        keys [j+1] = keys [j]
 keys [i] = y→keys [i].
 n += 1

void TreeNode :: remove (int k)
    idx = findkey (k)
    if (idx <n and keys[idx] == k
        if (leaf)
            removeFromLeaf (idx)
        else
            removeFromNonLeaf (idx)
    else
        if (leaf)
            cout << "key not polsent "
            return
        bool flag = (idx == n) ? true : false
        if child [idx] → n < 2
            fill (idx)
        if flag and idx > n
            child [idx -1] → remove (k)
        else child [idx] → remove[k]
```

Gurev Nanma

```
void TreeNode :: removeFromLeaf (int idx)
    for (i = idx+1  < n ; ++i)
        keys[i -1] = keys[i]
    n --
    return

void TreeNode :: removeFromNonLeaf (int idx)
    k = keys[idx]
    if (child[idx] -> n  >=2)
        pred  = getPred (idx)
        keys[idx] = pred
        child[idx] -> remove (pred)
    else if child[idx+1] -> n >= 2
        succ  = get Succ (idx)
        keys[idx] = succ
        child[idx + 1] -> remove (succ)
    else
        merge (idx)
        child [idx] -> remove (k)

void Tree :: remove (int k)
    if (!root) -> cout << " Tree is empty " return
    root -> remove (k)
    if (root -> n = = 0)  temp = root
        if (root -> leaf)  root = NULL
        else  root = root -> child [0]
        delete tmp
    return
```