Name - Harshit Hiremath
USN - 1BM18CS036

Lab6 : 2-3 Tree insertion & deletion

~~[illegible]~~ 5A
27/10/20

```
class Node {
    int *keys;
    Node **child;
    int n;
    bool leaf;
    friend class Tree;
}
class Tree {
    Node *root = NULL;
  public:
      void traverse(){
      if (root != NULL) root->traverse(); }
      void insert (int k){
      ~~void remove (int~~
          if (root==NULL){
             root=new Node(true);
             root -> keys[0]=k;
             root ->n= 1;
          }
          else {
             if(root->n ==3){
                Node *s = new ~~Tree~~ Node(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;
             }
```

[illegible]

Harshit
1BM18CS036

```
            else    root -> insertNonFull(k);
        }
    }

    void remove (int k) {
        if (!root) {
            cout << "Tree is empty" << endl;
            return;
        }
        root -> remove (k);
        if (root -> n == 0) {
            Node *tmp = root;
            if (root -> leaf)  root = NULL;
            else  root = root -> child [0];
            delete tmp;
        }
        return;
    }
}
void ~~Tree~~ Node :: insertNonFull (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 ++;
    }
    else {
        while (i >= 0 && keys [i] > k) i--;
        if (child [i+1] -> n == 3) {
            split Child (i+1, child (i+1]);
            if (keys [i+1] < k) i++;
        }
```

Harshit
1BM18CS036

```
            child[i+1] -> insertNonFull(k);
        }
}
void ~~Bree~~ Node :: splitChild (int i, ~~Bree~~ Node *y){
    Node *z = new Node(y->leaf;
    z->n=1;
    z-> keys[0] = y->keys[2]
    if (y-> leaf == false){
        for (int j=0; j<2; j++)
                z->child[j] = y ->child[j+2];
    }
    y->n=1;
    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++;
}
void   Node :: removeFromLeaf (int idx) {
    for ( int i =idx+1; i<n; ++i)
            keys[i-1] = keys[i];

    n--;
}
void Node :: removeFromNonleaf( int idx){
    int k= keys[idx];
    if (child[idx] ->n > 2){
        int  pred = getPred (idx );
        keys [idx] = pred;
        child [idx] -> remove (pred);
    }
```

[illegible]

Harshit
1BM18CS036

```
else if (child [idx +1] -> n >=2) {
    int succ= getsucc (idx);
    keys[idx] =succ;
    child [idx + 1] -> remove (succ);
  }
  else {
    merge (idx);
    child [idx] -> remove (k);
  }
}

void  TreeNode::remove (int  k) {
  int idx = findkey( k)
  if (idx <n && keys[idx] ==k) {
    if (leaf) removefromLeaf (idx);
    else  removeFromNonLeaf (idx);
  }
  else {
    if (leaf) {
      cout <<" key  doesn't exist" <<endl;
      return;
    }
    bool flag = ((idx==n)? true: false);
    if (child [idx] -> n < 2)
        fill (idx);
    if (flag && idx>n)  child [idx-1] ->remove (k);
    else  child [idx] ->remove (k);
  }
}
```

[illegible]