IBM19CS403
CHIRANJEEVI

2-3 tree insertion

```
class TreeNode
 {
   int   * keys;
   TreeNode  **child;
    int n;
    Bool  leaf;
   };

 class Tree
  {
    TreeNode   *root=NULL;

    Public:
     Valid traverse()
     {
       if (root != NULL)
            root -> traverse ();
       }
       void insert (int k);
       void remove (int k);
     };
void Tree : Insert (int k)
     {
       if (root == NULL)
        {
         root = new    TreeNode (true)
                root -> keys[0] = k;
                root -> n = 1;
        }
```

```
else
{
  if(root->n==3)
  {
   Tree Node *S= new  Tree Node(false);
      S-> child[0]= root;
      S-> Splitchild (0,root);
      int  i=0;
     if (s->keys[0]<k)
          i++;
     s-> child[i]-> insert  non full[k];
            root=s;
        }
     else
          root-> insert  nonfull(k);
     }
   }
void  Tree Node:: insert  Non full(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;
      }
```

chf.

```
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]-> insert  non full(k);
    }
  }

void   TreeNode:: Splitchild (int  i, TreeNod *y)
       {
        TreeNod   *z = new   TreeNode (y-left);
            z ->n =i
           z -> keys[0] = y-> keys [z];
             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 [i];
    child [i+1] = z;
 for(int  j=n-1; j>i; j--)
     keys [j+1] = keys[j];
     Keys[i]= y->keys [i];
          n = n+1;
    }

void   Tree Node :: remove (int k)
 {
   int    idx= find key(k)
          if (idx<n && keys[idx]==k)
          {
             if (leaf)
                    remove from leaf (idx);
             else
                  remove from Non leaf (idx);
          }
      else
        {
         if (leaf)
         {
         cout << " keys doesn't exist"<< ends;
                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);
          }
```

```
  return;
 }
void TreeNode :: remove from leaf (int idx)
        {
        for (int i=idx+1; i<n; i++)
           keys [i-1] = keys [i].
           n--;
        return;
     }

void Tree. Nod :: remove from non leaf (int idx)
   {
     int k = keys [idx];
       if (child [idx] -> n >= 2)
      {
       int pred = get_pred (idx);
         keys (idx) = pred;
         child [idx] -> remove (pred);

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

      }
```

chf.

```
else
{
    merge (idx);
    child [idx] -> remove (k);
    }
   return;
 }
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
             roo=root -> child [0];
                delete temp;
          }
      return;
    }
```