LAB-5 (2-3 TREE)

27/10/2020

REVANTH.R

1BM18CS082

[illegible]

```
class TreeNode
{  int  *key;
   TreeNode  ** child;
   int n ;
   bool  leaf;
};
                    class Tree
                  { TreeNode * root = NULL;
                    void 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      root → insert NonFull (k);  }
}
```

REVANTH · R
1BM18CS082

```
void TreeNode:: 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 = n+1 ;

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

Scanned with CamScanner

```
Void    TreeNode :: remove (int k)
{  int   x = findkey (k)
       if ( x < n && key [x] == k)
         { if (leaf)
            remove FromLeaf ( x );
       else  remove  from  NonLeaf (x) ;
       }    return ;
   }

  void      TreeNode :: remove FromLeaf ( int x)
   {   for   ( int i = x+1 ; i<n ; i++ )
             keys [i-1] = keys [i]
              n-- ;              return ;   }

   void    tree : remove   (int k)
 {   if ( ! root )    {   cout << " Tree  is empty " << endl;
        return ;      }
          root -> remove (k) ;
       }
          return ;
    }
```

[illegible]