```
B-Tree-Delete(k,x):
if k is not in node x:                          //this part is used to make sure all the
    if num_keys(x) > t or x is root:            nodes from root to the destination node
        continue                                (which k lies in) have at least t keys.
    else:                                       Therefore, when we later pop up or push
    y = left_im_sib(x);                         down the keys, it won't violate the
    z = right_im_sib(x);                        B-trees property. (Keys>t-1)
    w = root(x);
     if num_keys(y) >=t:                        // if one of the siblings is larger than t
        k'= most_right_key(y);                  keys, then use it through the root. That
        move(k',y,w);                           means pop up the nearest key in the
        k''= find_successor(k',w);              sibling to the root to hold the position.
        move(k",w,x);                           Then pushes down the nearest key in the
    elseif num_keys(z) >=t:                     root to the current node.(not exactly take
        k' = most_left_key(k,z);                the key in that sibling, but a key from
        move(k',z,w);                           the root)
        k''=find_predecess(k',w);
        move(k'',w,x);                          //if neither of the siblings has more or
    else:                                       equal than t keys, in the root, finds the
      if y:                                     key between the existed sibling and
          k'=find_key(y,x)                      current node. Push it down to current
          merge(k',y,x)                         node, and merge the node with this
      else:                                     sibling together.
          k'=find_key(z,x)                      //x.ci is the root of the subtree that must
          merge(k',z,x)                         contains k, this will recursively fill all
    B-Tree_Delete(k,x.ci)                       the nodes along the way.
    Return;

elseif k is in node x:                          //when k is in node x:
  if x is leaf:                                 If x is leaf:
    y = left_im_sib(x);
    z = right_im_sib(x);
    w=root(x);
    if num_keys(x) >= t:                        //Case1: it contains more than t keys,
        delete(k,x);                            just delete k.
    elseif num_keys(y) >=t:                     //Case2: one of its siblings contains
        k'= find_predecess(k,y);                more than t keys. The same idea with the
        move(k',y,root(x));                     above part, use it through the root. And
        k''= find_successor(k',w);              then current node will have t keys,
        move(k",w,x);                           delete the k.
        delete(k);
    elseif num_keys(z) >=t:
        k' = find_successor(k,z);
        move(k',z,w);
```

```
k''=find_predecess(k',w);                    //Case 3: if neither of the siblings has
        move(k'',w,x);                       more or equal than t keys, push down
        delete(k);                           the key in the root , between the current
    else:                                    rode and its exited sibling. Then merge
        if y:                                the current node with this sibling. Then
            k'=find_key(y,x)                 delete k
            move(k',w,x);
            x=merge(x,y);
            delete(k);
        else:
            k'=find_key(z,x)
           move(k',w,x);
            x=merge(x,z);
            delete(k);

else x is internode:                         //if x is internode
    y = left_im_child(x,k);
    z = right_im_child(x,k);                 //Case 4: if one o the children has
    if num_keys(y) >=t:                      more or eqal than    keys, just take the
        k'=find_predecess(k,y);              predecessor or successor in this node,
        move(k',y,x)                         Then dlete k.
        delete(k)
    elseif num_keys(z)>=t:
        k'=find_successor(k,z);              //Case 5: if noneof the children has
        move(k',z,x)                         enough keys o offer, push down the key
        delete(k,x)                          k to its child, and merge the two children
    else:                                    to form the newchild, then delete k.
        move(k,x,y);
        y=merge(y,z);
        delete(k,y);
```

Pseudo codes end here……

Functions:

left_im_sib(x)	get the left sibling of node x
move(x,m,n)	move the key x from node m to n
find_successor(k,z);	fin the successor of k in node z
find_key(x,y)	in the root, find the key which between its two children node: x and y.
num_keys(x)	give the number of keys in node x