```
Void decreasekeyBino(Node*H, int old-val, int new-vol)
{
 //Check element is Present or not.
 // Return if node is not present.
 // Reduce value to minimum.
 // update the heap according to reduced value.

 Node *node = findNode(H.old-val)
    if(node == Null)
        return;
    node -> val = new_val;
    node *parent = node -> parent;
    while(parent != Null && node->val < parent->val)
    {  swap(node -> val, parent -> val);
          node = parent;
          parent = parent -> parent;
    }
}
```

Scanned with CamScanner

```
// Function to Delete an Element from B.Heap.
Node * binoDelete (node *h, int val){
// check if heap is empty or not.
// Reduce value to minimum.
// Delete minimum element from Bheap
 if(n == NULL)
     return NULL;
 decreasekeyBino(n, val, Int_min);
 return extralmin(n);
}

// Find node

Node *FindNode(node *h, int val){
 if(n == NULL)
     return NULL;
 if(n->val  ==  val)
   return h;
 Node *res = findNode(h->child, val);
    if(res != NULL)
        return res;
    return findnode(h->sibling, val);
}
```