

void decreaseKey(B: no(Node *H, int old-val, int new-val)

{ // 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;

}

}

// 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 (h == NULL)  
        return NULL;  
    decreaseKeyBino(h, val, INT_MIN);  
    return extractMin(h);  
}
```

// Find node

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