AKSHAY MITTUR
1BM18CS010

16/12/20

ADS LAB
BINOMIAL HEAP WRITEUP

```
function delete(~~{~~ Node *h, int val) {
        if(!h) return NULL;
        decreasekeyBHeap(h, val, INT_MIN);
        return extractMinHeap(h);
}
function decreasekeyBHeap(Node *H, int oldv, int newv){
        Node* node = findNode(H, oldv);
        if(!node) return;
        node->val = newv;
        Node *parent = node->parent;
        while(parent != NULL && node->val < parent->val) {
                swap(node->val, parent->val);
                node=parent;
                parent= parent->parent;
        }
}

function *extractMinHeap(Node *h) {
        if (!h) return NULL;
        Node *min_prev= NULL;
        Node * min_=h;
        int min= h->val;
        node *curr=h;
        While (curr->sibling != NULL) {
                if ((curr->sibling)->val <min ){
                        Min = curr->sibling->val;
                        Min_prev= curr;
                        min_= curr->sibling;
                }
```

AKSHAY MITTUR
1BM18CS010

classmate
Date
Page

16/12/20

ADS LAB

```
        curr=curr->sibling;
    }
    if(min_prev==NULL && min->sibling==NULL) h=NULL;
    else if(min_prev==NULL) h=min->sibling;
    else min_prev->sibling= min->sibling;

    if(min->child){
            revertlist(min->child);
            min->child->sibling=NULL;
    }
    return unionBHeap(h,root);
}
function findNode(Node *h, int val){
        if(!h) return NULL;
        if(h->val ==val) return h;
        Node *res= findNode(h->child, val);
        if(res!=NULL) return res;
        return findNode(h->sibling,val);
}
function revertlist(Node *h){
        if(h->sibling) {
                revertlist(h->sibling);
                h->sibling->sibling=h;
        } else root=h;
}
```