Prathik Karanth
1BM18CS139

ADS Lab
Binomial Heap

```
func delete ( Node *H, int val) {
        if (!H) return NULL;
        decrease key BHeap ( h, val, INT_MIN);
        return extractMinHeap (h);
}
func  decrease key BHeap (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;
              m = curr->sibling;
            }
```

```
        curr = curr -> sibling;
    }
    if (min_prev == NULL && min -> sibling == NULL)
    else if (min_prev == NULL) h = min -> sibling;
    else min_prev -> sibling = min -> sibling;
    if (min -> child) {
            reverselist (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 reverselist (Node *h){
        if (h -> sibling) {
                reverselist (h -> sibling);
                h -> sibling -> sibling = h;
        } else root = h;
}
```