S.R.PUNEETH
1BM18CS087
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;
```


S.R.Puneeth

S.R.PUNEETH
1BM18CS087

```
while (curr -> sibling != NULL)
{
if ((curr -> sibling) -> val < min) {
  min = curr -> sibling -> val;
  min_prev = curr;
  min_ = curr -> sibling;
}
  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) {
  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;
}
```


S.R.Puneeth