Binomial Heap Delete

Krishnaprasad V
1AH18CS077
S1

```
func delete ( Node *h, int val)
{     if (!h) return NULL
      decreaseKeyBHeap (h, val, INT_MIN)
      return extractMinHeap (h)
}
func decreaseKeyBHeap (Node * H, int oldv, int newv)
{     Node * node = findNode(H, oldv)
      if (!node) return
      node -> id = NEWV;
      Node *parent = Node -> parent;
      while (parent != NULL && node -> val < parent->val)
      {  swap (node->val, parent->val)
         node = parent;
         parent = parent->parent
      }
}
func     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
    }
  }
  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   merge Heap (h, root)
}
func   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)
}
func   revertList (Node *h)
 {  if (!h -> sibling)  { revertList(h -> sibling); h -> sibling -> sibling = h; }
    else   root = h
}
```