Decrease key in Bheap

Kshitij R
1BM18CS048

Lab-10 writeup

```
void decreasekeyBheap (Node *h, int old_val, int new_val)
{  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;
   }
}

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

```
Node * binodelete(Node *h, int val)
{ if(h == NULL)
        return NULL;
    decreaseKeyBheap(h, val, INT_MIN);
    return extractheap(h);
}
Node * ExtractMnBheap(Node *h)
{   if(h == NULL)
        return NULL;
    Node * min_node_prev = NULL;
    Node * min_node = h;
    int min = h->val;
    Node * curr = h;
    while(curr->sibling) -> val < min)
    { if (curr->sibling != NULL)
      { min = ((curr->sibling)->val;
        min_node_prev = curr;
        min_node = curr->sibling;
      }
    curr = curr->sibling;
    if(min_node_prev == NULL && min_node->sibling == NULL)
      h = NULL;
    else if (min_node_prev == NULL)
        h = min_node->sibling;
    else
```

```
min_node_prev -> sibling = min_node -> sibling;
if(min_node -> child != NULL)
{
  reverseList(min_Node -> child);
  (min_node -> child) -> sibling = NULL;
}
return unionHeaps(h, root);
}
```