K.V. Sravya

Kattirisetty Venkata Sravya
1BM18CS041H

```
using namespace std;
struct Node
{
    int val, degree;
    Node *parent, *child, *sibling;
};

Node* Delete(Node *head, int val)
{
    if (head == NULL)
        return NULL;
    decreasekey(head, val, INT_MIN);
    return MinHeap(head);
}

void decreasekey(Node *head, int o, int n)
{
    Node *node = findNode(head, o);
    if (node == NULL)
        return;
    node->val = n;
    Node *parent = node->parent;
    while (parent != NULL && node->val < parent->val)
    {
        swap(node->val, parent->val);
        node = parent;
        parent = parent->parent;
    }
```

K.V.Sravya

Kattiselty Venkata
Sravya
1BM18CS044

```
Node minHeap(Node *h)
{ if (h==NULL)
   return NULL;
  Node * mprev = NULL;
  Node * mn = h;
  int m = h->val;
  Node *cur = h;
  while (cur->sibling != NULL)
  {
    if ((cur->sibling)->val < m)
     {
      min = (cur->sibling)->val;
      m_prev = cur;
      m_n = cur->sibling;
     }
    cur = cur->sibling;
  }
 if (m_prev == NULL && m_n->sibling == NULL)
       h=NULL;
 else if (m_prev == NULL)
     h = m_n->sibling;
 else   m_prev->sibling = m_n->sibling;
 if (m_n->child != NULL)
    { reverse(m_n->child);
      (m_n->child)->sibling =NULL;
    }
    return union(h, root);
}
```

K.V.Sravya

Katturisetty Venkata Sravya

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

Node *union (Node *head1, Node *head2)
{ if (head1 == NULL && head2 == NULL)
    return NULL;
  Node *res = merge (head1, head2)
  Node *p = NULL, *c = res, *n = c->sibling;
  while (n != NULL)
  { if (c->degree != n->degree) || (n->sibling != NULL) &&
                                   (n->sibling)->degree ==
                                   c->degree))
    {
      p = c; c = n;
    }
    else { if (c->val == n->val)
           { c->sibling = n->sibling;
           }
         else { if (p == NULL)
                  res = n;
                else {p->Sibling = n;  c = n;  }
              n = c->sibling; }
  return res;
}
```