M. S. SATHWICK KIRAN

1BM18CS050

ADS LAB-10

⟹ Binomial Heap (Delete) :-

```
function delete (Node* h, int val)
  {
     if (! h) return NULL;
     decreaseKey BHeap (h, val, INT_MIN);
     return   extract Min Heap (h);
  }
     function decreaseKey BHeap (Node * H, int old, int new V)
      {
          Node * node = find Node (H, old);
          if (! node) return;
           Node → val = new V;
           Node * parent = node → parent;
        while (parent = NULL && node → val < parent → val) {
     swap ~~heap~~ (node → val, parent → val);
           Node = parent;
              parent = parent → parent;
         }
       }
             function   extract Min Heap (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) {
     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  revert list (Node *h){
          if (h → sibling){
              revertList (h → sibling);
              h → sibling → sibling = h; }
                  else root = h;
          }
```