PRANAV JAGADEESH 1BM18CS071 5 B BATCH 4 PROGRAM 10

1. delete(H): Like Binary Heap, delete operation first reduces the key to minus infinite, then calls extractMin()
2. decreaseKey(H): decreaseKey() is also similar to Binary Heap. We compare the decreases key with it parent and if parent's key is more, we swap keys and recur for parent. We stop when we either reach a node whose parent has smaller key or we hit the root node.

```
struct Node
{  int val, degree
   Node *parent, *child, *sibling
};
Node *root ← NULL
int binomialLink (Node *h1, Node *h2)
{  h1 → parent ← h2
   h2 → sibling ← h2 → child
   h2 → child ← h1
   h2 → degree ← h2 → degree + 1
}
```

```
Node *createNode(int n)
{ Node *new_node = new Node
  new_node->val = n
  new_node->parent = NULL
  new_node->sibling = NULL
  new_node->child = NULL
  new_node->degree = 0
  return new_node
}

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 *binomialHeapDelete(Node *h, int val)
{ if (h == NULL)
   return NULL
  decreaseKeyBHeap(h, val, INT_MIN)
  return extractMinBHeap(h)
}
```