Binomial Heap :- Decrease Key (H)
Delete (H)

P. SAI DEEKSHITH
1BM18CS148
5th Sem 'B'

```
Void  decreasekeyBin (Node * H, int old-val,
                                 int new-val)
{
      // 1. Check element is present or not
      // 2. Return if node is not present
      // 3. Reduce value to Minimum
      // 4. update the heap according
      // ~~value~~ to the reduced Value.

   Node * node = find Node (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;
}     }
```

```
// Funtion    Delete an Element from B Heap

Node* binDelete (Node* h, int val){

    // 1. check if heap is empty or not
    // 2. Reduce value to minimum
    // 3. Delete min Element from B Heap

    if (h == NULL)
            return NULL;

    decrease keyBino(h.val, INT_MIN);

    return extractMin (h);

}


// Function    Find Nod
Node *FindNode(Node *h, int val){
        if (h == NULL)
            return NULL;

        if (h ->val == val)
            return h;

    Node* res = findNode(h->child, val);
        if (res != NULL) return res;

    return findNode(h-> sibling, val);

}
```