Aubay Ahmed
1BM19CS401

Binomial heap
(i) union (H1, H2)
(ii) getMin(H)
(iii) extractMin(H)

```
struct node
{
    int data;
    Node *child, *sibling, *parent;
    int degree;
};
Node *newNode(int key)
{
    Node *temp = new Node;
    temp->data = key;
    temp->degree = 0;
    temp->child = temp->parent = temp->sibling = NULL;
    return temp;
}
int <Node*> union (list<Node*> head, int key)
{
    Node *temp = newNode(key);
    return unionBinHeap(head, temp);
}
```

```
Node* getMin (list<Node*> heap)
{ list<Node*>::iterator it = heap.begin();
  Node *temp = *it;
  while (it != heap.end()) {
      if (*it)->date < temp->date)
          temp = *it;
          it++;
      }
      return temp;
  }

list<Node*> removeMin (list<Node*> heap)
{ list<Node*> newNodeList;
  Node *temp
  temp = getMin (heap);
  list<Node*>::iterator it;
    it = head.begin()
  while (it != head.end()) {
      if (*it != temp)
          newHeap.push_back (*it);
          it++;
  }
```

```
lo = removeNodeFromTreeReturnBigHeap(temp);
newheap = unionTwoHeaps(newheap, lo);
newheap.adjust(newheap);
return newheap;

void printTree(Node *h)
{ while(h){
  cout << h->data << " ";
  printTree(h->child)
  h = h->sibling;
}}

void printHeap(list<Node*> heap){
  list<Node*>::iterator it;
  it = heap.begin();
  while(it != heap.end())
  {
    printTree(*it)
    it++;
}}
```