```
Insert (list _head, int key) {
    Node *temp = NewNode(key);
    return insertATreeInHeap(_head, temp);
}

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

list<Node*> extractMin(list _heap) {
    list<Node*> new_heap, lo;
    Node *temp;
    temp = getMin(_heap);
    it = _heap.begin();
    while (it != _heap.end()) {
        if (*it != temp)
            new_heap.push_back(*it);
        it++;
    }
    lo = removeMinFromTreeReturnBHeap(temp);
    new_heap = unionBinomialHeap(new_heap, lo);
    new_heap = adjust(new_heap)
    return new_heap;
}
```