P SAI DEEKSHITH

Binomial Heap :-

```
list<Node*> insert (list<Node*> _head, int key)
{
    Node * temp = newNode(key)
    return insertATree InHeap (_head, temp);
}

list<Node*> insertATree InHeap (list<Node*> _heap,
                                Node * tree)
{
    list<Node*> temp;
    temp.push_back (tree);
    temp = unionBinomialHeap(_heap, temp);
    return adjust (temp);
}
```

Scanned with CamScanner

```
Node* getMin(list<Node*> _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<Node*> _heap)
{
    list<Node*> new_heap, it;
    Node* temp;
    temp = getMin(_heap);
    list<Node*>::iterator it;
    it = _heap.begin();
    while(it != _heap.end())
    {
        if(*it != temp)
            new_heap.push_back(*it);
        it++;
    }
```

Scanned with CamScanner

```
to = removeMinFromTreeReturnBHeap(temp);
new_heap = unionBinomial(Heap(new_heap, to));
new_heap = adjust(new_heap);
return new_heap;
}
```

Scanned with CamScanner