Binomial Heap

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

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, lo;
    Node *temp;
    temp = getMin (_heap);
    list <Node*> :: iterator it;
    it = _heap.begin ();
    while (it != _heap.end () )
    {
        if (it != temp)
        {
            ~~new_temp~~ new_heap.push_back (it);
        }
        it ++
    }
    lo = removeMinFromTreeReturnBheap (temp);
    new_heap = ~~adjust~~ unionBionomialHeap (New_heap_lo);
    new_heap = adjust (new_heap);
    return new_heap;
}.
```