Harshit Hiremath,
IBM18CS036

09/12/20

ADS - Lab 10

Binomial Heap

Utility functions :-

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

list<Node*> unionBinomialHeap(list<Node*> l1, list<Node*> l2){
        list<Node*> new;
        list<Node*>:: iterator ot= l2.begin();
        list<Node*>:: iterator it = l1.begin();
        while ( it != l1.end() && ot != l2.end()) {
            if((*it)->degree <= (*ot)->degree) {
                new.push_back(*it);
                it++;
            }
            else{
                new.push_back(*ot);
                ot++;
            }
        }
        while (it != l1.end()){
            new.push_back(*it);
            it++;
        }
        while(ot != l2.end()){
            new.push_back(*ot);
            ot++;
        }
        return new;
}
```

Insert, getMin, & extractMin functions

```
list<Node*> insert(list<Node*> head, int key){
    Node *temp = newNode(key);
    return insert4TreeInHeap(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_heap.push_back(*it);
        }
        it++;
    }
    lo = removeMinfromTreeReturnBHeap(temp);
    new_heap = unionBiromialHeap(new_heap, lo);
    new_heap = adjust(new_heap);
    return new_heap;
}
```