ADS LAB
Khaithik. S. Anand
1BM18CS046

papergrid
Date: / /

Binomial Heap

Insert Function : (input : head, key)
{

```
Node *temp = newNode(key);
list<Node*> t;
t.push_back(temp);
t = unionBH(head H)
return adjust(t);
}

adjust (list<Node*> heap){
    if (heap.size <=1) return heap;
    list<Node*> newHeap.
    auto i+1, i+2, i+3;
    i+1 = i+2 = i+3 = heap.begin();
    if (heap.size() == 2)
    {
        i+2 = i+1,
        i+2++;
        i+3 = heap.end();
    } else {
        i+2++;
        i+3 = i+2;
        i+3++;
    } while(i+1 != heap.end()) {
        if (i+2 == heap.end()) i+1++;
        else if(i+1->degree < i+2->degree) {
            i++, i+2++.
            if (i+3 != heap.end()) i+3++;
        }
    }
```

Karthik .S. Anand
1BM18CS026

Date: / /

```
else if(*i+1->degree == *i+2->degree){
        Node *temp;
        *i+1 = merge(*i+1, *i+2);
        i+2 = heap.erase(i+2);
        if(i+3 != heap.end()) i+3--;
}
else if(i+3 != heap.end() && *i+1->degree == *i+2->degree
        && *i->degree == *i+3->degree){
    *i++, *i+2++, i+3++;
}
    return heap;
}
Function Getmin(list<Node*> heap){
        auto it heap.begin();
        while(it != heap.end()){
            if(*it->data < temp->data) temp = *it;
            i+++;
        }
        return heap
    }
Function extract min(list<Node*heap){
        list<Node*> new_heap, to, Node *temp;
        temp = getmin(heap), auto it = heap.begin();
        while(it != heap.end()){
            if(*it != temp) new.heappush_back(*it);
            i+++;
        }
        to = reverse(temp);
        new_heap = unionBH(new_heap, to)
        new_heap = adjust(new_heap);
        return new_heap;
}
```