Date __/__/__
Page
SPLASH

Perauth.NM
1BM18CS081

Binomial heap insert

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

list <Node *) remove (Node * tree ) {
    list <Nod *> heap;
    Node * temp = tree -> child;
    Node * lo;
    while (temp) {
        lo = temp;
        temp = temp -> sibling;
        lo -> sibling = NULL;
        heap.push_front (lo);
    }
    return heap;
}

list <Node *> insert ( list <Node*> head , int key) {
    Node * temp = new Node (key);
    return insertHeap ( head, temp);
}

Node getMin ( list <Node*> heap) {
    list <Node *) iterator it = heap.begin();
    Node * temp = * it;
```

Date __/__/__
Page
SPLASH

```
while (it != heap.end()){
        if (*it -> data < temp -> data)
            temp = *it;
        it++;
    }
    return temp;
}

temp = getmin(-heap);
list<Node*>::  it = heap.begin()
while (it != heap.end()){
    if (it != temp){
        newheap.push_back(*it);
        it++;
    }
    h = unionminfromtree returnBHeap(temp);
    newheap = unionBinomialheap(newheap h);
    newheap = adjust(newheap);
    return newheap;
}
```

Rekauth.NM