Name: Hemanth Kumar V.C
USN: 1BM18CS037

Date:
Page No:

Insert function - inserting a key into the binomial heap

insert (list<Node*>-head, int key)
- create new node temp
- call insert to heap function

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

list<Node*> removeMinFromTreeReturnBHeap(Node*tree)
{
    list<Node*> heap;
    Node *temp = temp -> 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 = 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_heap.push_back(*it);
    }
    it++;
```

```
lo = removeMinFromTree ReturnBHeap(temp);
new_heap = UnionBinomialHeap(new_heap, lo);
new_heap = adjust(new_heap);
return new_heap;
}

void printTree(node *h)
{
    while(h)
    {
        cout<<h->data<<" ";
        printTree(h->child);
        h=h->sibling;
    }
}
```