9] Implementation of Binomial Heap.

i) Insert Function : inserting a key into binomial heap.

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

list <Node *> removeMinFromTree(Node *tree)
{
   list <Node *> 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 insertATreeinHeap(_head, temp);
}
```

Chinag Swamy
1BM16CS409

ii)

```
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;
```

iii)

```
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(temp);
    new_heap = unionBinomialHeap(new_heap, lo);
    new_heap = adjust(new_heap);
    return new_heap;
}
```