S.R.PUNEETH
1BM18CS087

ADS LAB 10 - Binomial Heap

Insert Function: (input: head, key)

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

```
adjust(list<Node *> heap)
{
 if(heap.size <= 1)
 return heap;
 list<Node *> new_heap;
 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++;
 }
```


S.R.Puneeth.

S.R.PUNEETH
1BM18CS087

```
while(i+1 != heap.end()) {
  if(i+2 == heap.end())
   i+1++;
  else if(i+1 -> degree < i+2 -> degree) {
   i+1++, i+2++;
   if(i+3 != heap.end())
   i+3++;
  }
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;
}

FunctionGetMin(list<Node *> heap){
          auto it= heap.begin();
          while(i+1 = heap.end()) {
        if(*it -> data < temp -> data)
          temp = +it; it++;
        }
```

S.R.PUNEETH
1BM18CS087

```
return temp;
}

Function extractMin( list < Node * temp )
{
 list < Node * > new_heap, lo, Node * temp;
 temp = getMin(heap), auto it= heap.begin();
 while( it != heap.end() ) {
   if (*it != temp )
     new_heap push_back(*it);
     it++;
 }
 lo = revn(temp);
 new_heap = unionBH (new_heap, lo);
 new_heap = adjust (new_heap);
 return new_heap;
}
```