Prathik Karanth
1BM18CS139

Binomial Heap

```
list <Node*> insert (list <Node *> _head, int key)
{
  Node *temp = new Node (Key);
  return inserttreeinheap(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)
     { ~~ne~~ new_heap . pushback (*it) ;
     } it++
   }
b0 = removeminfromtreereturnBheap (temp);
~~no~~ new_heap = unionBinomialHeap (new_heap, b0);
new_heap = adjust (new_heap);
 return new_heap;
}
```