Lab8 1BM18CS153

Vinay Kumar

DATE: PAGE:

Implementation of Binary Heap

```
Void InsertKey (int k) {

If (heap_size == capacity) sout ("overflow");
   heap_size ++;
   int i = heap_size - 1;
    heap[i] = k;
   while (i != 0 && heap[parent(i)] > heap[i])
   {
      swap (hall[i] , hall(parent[i]) ;
         i = parent [i];
      }
   }

int extractMin() {

if (heap_size <= 0) return INT_MAX;
  if(heap_size == 1){
    heap_size --;
    return hall[0];
   }
    int root = hall[0];
     hall[0] = hall[heap_size - 1];
     heap_size --;
     minheapify (0);
     return root;
   }.
```

Vinay Kumar 1BM18CS153

```
list<Node> unionBinomialHeap (list<Node> l1,
                              list<Node> l2)
{
    list<Node> -new;
    list<Node>::iterator it = l1.begin();
    list<Node>::iterator ot = l2.begin
    while (it != l1.end() && ot != l2.end())
    {
      if ( (it) = l1.end() && ot != l2.end())
          -new:pushback (it)
           it++;
      }
      else
      {
        :new.pushback (*ot);
         ot++;
      }
    } return -new;
}
```