Raghav Mittal
2BM18CS075

Program - 9 Heap

Insert

```
list <Node *> insert a tree in Heap (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 = 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 = newNode (key);
```

return insert a tree heap (_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;

}

temp = getmin (_heap);

list <Node*> ::

it = _heap.begin();

while (it != _heap.end ()) {

if (it != temp) {

new_heap.push_back (it); }

it ++. }

h = union rem from tree Return B Heap (temp);

newheap = UnionBinomialHeap (new_heap, h);

new_heap = adjust (new_heap);

return new_heap;

}.