Solutions to Re-exam 2003 Assignments

Exercise 2

1. Idea: Extract in parallel one by one integers from both queues and compare if the extracted integer from one queue is equal to the extracted integer from the other queue. Also, you have to skip duplicates (lines 5–8), and make sure that if one queue is emptied, another one is empty too (line 9). The worst-case complexity of the algorithm is $O(n \lg n)$, where n is the number of elements in the largest of the two queues.

```
EQUALSETS($q1, q2$:PriorityQueue)
 1   top1 ← q1.extractMax()
 2   top2 ← q2.extractMax()
 3   while top1 ≠ NIL and top2 ≠ NIL do
 4       if top1 ≠ top2 then return false
 5       do ntop ← q1.extractMax() while ntop = top1
 6       top1 ← ntop
 7       do ntop ← q2.extractMax() while ntop = top2
 8       top2 ← ntop
 9   if top1 ≠ NIL or top2 ≠ NIL then return false        ▷ One is a subset of another
10   return true
```

2. This algorithm assumes that the argument a is not NIL. A standard queue data structure is used to implement a breadth first traversal of the tree. Note that the loop in line 3 is not infinite, as long as the tree is valid. The worst-case complexity of the algorithm is $O(n)$, where n is the number of nodes in the tree.

```
FINDPLACE($a$:BinTree)
 1   q:Queue
 2   q.enqueue(a)
 3   while true do
 4       x ← q.dequeue()
 5       if x.leftSubtree() = NIL then return x
 6       else q.enqueue(x.leftSubtree())
 7       if x.rightSubtree() = NIL then return x
 8       else q.enqueue(x.rightSubtree())
```

3. Note that, as stated in the exercise, h is the max-heap (an instance of the BINTREE ADT), corresponding to the priority queue where we want to insert v.

```
INSERT($v$:int)
    ch ← findPlace(h).addNewChild(v)
    while ch.parent() ≠ NIL and ch.root() > ch.parent().root() do        ▷ "bubble up"
        r ← ch.root()
        ch.setRoot(ch.parent().root())
        ch.parent().setRoot(r)
        ch ← ch.parent()
```