Exercise 2

1. The scheme is an instance of the general quadratic probing scheme with constants $c_1 = 1$ and $c_2 = 0$
   This reduces the scheme to a general linear probing scheme
2. The probing sequence is permutation of this because only the complete table can be probed once before the condition m != j is not satisfied anymore. Therefore you cannot generate more elements than in the set and only every single one once.

   The cases are:

   1. Case: Table entry is empty, instant insertion (same sequence)
   2. Case: Table entry is not empty skip I values til empty list place (permutation with different beginning element)
   3. Case: Table is full: Termination, no insertion (permutation with different beginning element)

Exercise 3

Subtask a)

Insert(key, value):

```
    insert(key, value)
```

Maximum():

```
    node = root
    while node.right != null:
        node = node.right
    return node.value
```

Extract-max():

```
    value = Maximum()
    node = search(value)
    delete(node)
    return value
```

Increase-key(key):

```
    node = search(key)
```

priority, value = node.key + 1, node.value

delete(node)

insert(node(priority, value))

return priority

Decrease-key(key):

node = search(key)

priority, value = node.key - 1, node.value

delete(node)

insert(node(priority, value))

return priority

Subtask b)

	AVL-Tree	**Heap**
insertion	$O(\log n)$	$O(\log n)$
maximum	$O(\log n)$	$O(\log n)$
extract-max	$O(\log n)$	$O(\log n)$
increase-key	$O(\log n)$	$O(\log n)$
decrease-key	$O(\log n)$	$O(\log n)$

Seem to have the same complexity