Name:

Algorithms and Data Structures
Jacobs University Bremen
Dr. Florian Rabe

Quiz 2
given: 2017-03-07

You have 20 minutes.

Problem 1

Points: 2+2+3+3

Consider the following algorithm:

```
fun foo(l : List[int], m : List[int]) : List[Int] =
    x := l
    y := m
    while y ≠ []
        x := append(x, y.head)
        y := y.tail
    return x
```

List is implemented as an immutable linked list, and *append* adds an element at the end of a list.

1. Which operation does foo implement?
2. Assuming that both inputs have the same length n, give the Θ-class of the time complexity of foo in terms of n?
3. Give a loop invariant $F(l, m, x, y)$ for the while-loop with which we can prove foo partially correct.
4. Argue informally why the while-loop terminates (2 points), **OR** give a formal termination ordering $T(l, m, x, y)$ for the while-loop (3 points).

Solution:

1. $concat(l, m)$
2. $\Theta(n^2)$ (the problem is linear but the implementation is inefficient because each call to *append* takes linear time; prepending the elements of x in reverse order to y would be linear)
3. $concat(x, y) = concat(l, m)$
   To prove partial correctness, we need three things:
   - The loop invariant is true before the loop — trivial.
   - The formula is indeed a loop invariant — true because each iteration of the loop shifts the first element of y to the end of x
   - The needed property (return $concat(l, m)$) follows from the loop invariant and the negated loop condition — trivial.
4. It terminates because y becomes shorter in each iteration of the loop. A termination ordering is $length(y)$.

Problem 2

Points: 2+2+3+3

Consider the following algorithm operating on a mutable list x:

```
fun foo(x : List[int]) =
    low := 0
    high := length(x) − 1
    while low < high
        l := get(x, low)
        h := get(x, high)
        update(x, low, h)
        update(x, high, l)
        low := low + 1
        high := high − 1
```

Name:

1. Which operation does foo implement?
2. Assuming that *List* is implemented as an array, give the Θ-class of the time complexity of foo (in terms of the length n of x).
3. Assuming that *List* is implemented as a linked list, give the Θ-class of the time complexity of foo (in terms of the length n of x).
4. Argue informally why the while-loop terminates (2 points), **OR** give a formal termination ordering $T(x, low, high)$ for the while-loop (3 points).

Solution:

1. It reverts x in place.
2. $\Theta(n)$
3. $\Theta(n^2)$ (because *get* and *update* now take linear time)
4. It terminates because low increases and $high$ decreases; eventually $low < high$ must become false. A termination ordering is $high - low + 1$. $high + 1$ or $length(x) - low$ are also termination orderings. (The $+1$ are needed to make sure the expression never becomes negative.)
