Name: ______________________________

Algorithms and Data Structures
Jacobs University Bremen
Dr. Florian Rabe

Quiz 3
given: 2017-03-16

You have 20 minutes.

Problem 1

Points: 2+2+2+2

Give the Θ-class of the **average**-case time complexity (in terms of the length of the list) of the following sorting algorithms:

1. bubblesort
2. quicksort
3. mergesort
4. a variant of mergesort that splits the list into 3 sublists instead of 2

Solution:

1. $\Theta(n^2)$
2. $\Theta(n \log_2 n)$ (worst-case would be n^2)
3. $\Theta(n \log_2 n)$
4. $\Theta(n \log_3 n)$ (which is equal to $\Theta(n \log_2 n)$)

Problem 2

Points: 3+3

Consider quicksort.

1. Why is it essential to choose a good pivot element?
2. Why is it difficult to choose a good pivot element?

Solution:

1. Because the choice of pivot determines the time complexity. Always choosing the median yields $\Theta(n \log n)$, always choosing the smallest or greatest element yields $\Theta(n^2)$.
2. Because finding the median (the optimal choice) takes linear time itself, and that would yield $\Theta(n^2)$ again. quicksort works only if we choose the pivot in constant time.

Problem 3

Points: 3+3

Consider the following algorithm for sorting lists of natural numbers:

```
fun $foosort(l : List[\mathbb{N}]) : List[\mathbb{N}]$ =
    if ($l == []$) {return []}
    $g := ge(l)$                                    $ge(l)$ returns greatest element of $l$
    $count := Array[\mathbb{N}](g+1)$               new array with elements $count[0], \ldots, count[g]$
    for $i$ from $0$ to $g$
        $count[i] := 0$
    for $i$ from $0$ to $length(l) - 1$
        $count[l[i]] := count[l[i]] + 1$
    $r := []$
    for $i$ from $0$ to $g$
        for $j$ from $1$ to $count[i]$
            $r := prepend(i, r)$
    return $reverse(r)$
```

Name: ____

Regarding the efficiency of foosort, give its key

1. advantage
2. drawback

Solution: This algorithm is also called countingsort.

1. Its time complexity is linear in $length(l)$, which is much better than other sorting algorithms.
2. Its space complexity is $\Theta(ge(l))$, which is prohibitively expensive if $ge(l)$ is large.

Note that, in general, the time complexity is at least the space complexity under the reasonable assumption that all allocated memory is used at least once. The time complexity of $foosort$ is $\Theta(ge(l) + length(l))$.