CSC 225 (Fall 2016): Algorithms and Data Structures I
Assignment 2—theoretical part

Question 1.

a) In pseudo code, describe a *recursive* algorithm that reverses the elements in a singly linked list.

b) Describe the running time of your algorithm in a) as recurrence equation. Justify your answer.

c) Determine big-oh of your algorithm in a) by determining the closed form of your recurrence equation in b). Show your work using the repeated substitution method.

Question 2.

a) In pseudo code, describe an iterative algorithm that reverses the elements in a singly linked list.

b) Describe, in general, how to convert a given recursive algorithm into an iterative algorithm.

Question 3.

Let array A be an array that contains $n - 1$ unique integers in the range $[0, n - 1]$. That is, there is one number in range $[0, n - 1]$ that is *not* stored in A. Describe in pseudo-code an $O(n)$-time algorithm for finding that number. You are only allowed to use $O(log\ n)$ bits of additional space besides the array A itself.

Question 4.

Consider the following so called *Master Theorem* for recurrence equations. It applies to recurrence equations of the following form $\mathrm{T}(n) = a\mathrm{T}(n/b) + c\ n^d$, where:

- $\mathrm{T}(n)$ is an increasing function,
- $a \geq 1$ is a constant,
- $b > 1$ is an integer constant,
- $c > 0$ is a constant, and
- $d \geq 0$ is a constant.

Then $T(n)$ is

1. $O(n^d)$ if $a < b^d$,

2. $O(n^d \log n)$ if $a = b^d$, and

3. $O(n^{\log_b a})$ if $a > b^d$.

Consider the following algorithm.

```
Algorithm FindMin(A, front, last)
    if last-front $\leq$ 1 then return A[front]
    else
        midpoint $\leftarrow \lfloor$(front+last)/2$\rfloor$
        $min_f \leftarrow$ FindMin(A, front, midpoint)
        $min_l \leftarrow$ FindMin(A, midpoint+1, last)
        return min{$min_f$, $min_l$}
    end
```

a) What is the recurrence equation describing the running time of Algorithm FindMin?

b) Does this recurrence equation fit the Master Theorem? If so, what are a, b, c and d?

c) What is the Big-Oh running time of Algorithm FindMin?

Question 5.

Consider the sorting algorithms *insertion sort* and *merge sort*. How many *comparisons* will each of the two algorithms execute for the following instances (assume that you are to sort the elements in increasing order):

a) 1 2 3 4 5

b) 5 4 3 2 1

c) For insertion sort, determine the big-Oh of the worst-case number of comparisons for a sequence of n elements. Justify your answer.