apl. Prof. Dr. Thomas Noll
Sebastian Junges, Christoph Matheja, Matthias Volk

Compiler Construction 2017
— Exercise Sheet 1 —

Hand in until May 16th before the exercise class.

General Remarks

- You may hand in your solutions for the theoretical part of the exercises just before the exercise class starts *or* by dropping them into the "Compiler Construction" box at our chair. Do *not* hand in your solutions for the theoretical part via L2P.
- We do *not* insist on handwritten solutions.
- There also is a practical programming exercise published on a separate exercise sheet.
- We will publish solutions in the L2P.
- If you are still looking for a group or your group has less than 3 members, please post in the L2P forum.
- If you have questions regarding the exercises and/or lecture, feel free to post in the L2P forum, write us an email at cb2017@i2.informatik.rwth-aachen.de or visit us at our office.

Exercise 1 (2 Points)

As an example of the language *WHILE*, consider the following program.

```
1  /* A random walk */
2  int x = 10;
3  int s = 0;
4  while ( x > 0 ) {
5      int b = read() % 2; // randomness by user input
6      if (b == 1) {
7          x = x + 1;
8      } else {
9          x = x - 1;
10     }
11     s++;
12 }
13 write("I stopped walking after: ");
14 write(s);
15 write(" steps");
```

(a) Give a complete list of the symbol classes and corresponding tokens needed for the lexical analysis of our programming language *WHILE*. Recall that *WHILE* captures (integer) variable declarations, assignments, arithmetic operations, conditional branches, loops, basic I/O (read and write) and Java-style comments.

(b) Decompose the if-statement (lines 6-10) into a sequence of lexemes and translate each lexeme into a symbol.

Exercise 2 (3 Points)

Let Ω be the set of all relevant characters in some encoding, e.g. all UTF-8 characters.

(a) Provide regular expressions for comments (`//` and `/* ... */`) in *WHILE*. Please keep in mind that `*` as well as `/` may also occur inside of comments. Thus `/*foo * bar / */` is a valid comment. Note that single-line comments are terminated by a newline symbol `\n` (Linux), carriage return `\r` (Mac OS) or `\r\n` (Windows).

(b) Provide a regular expression capturing numbers in scientific notation, e.g. `-17.42e+23`. To be more precise, a number in scientific notation consists of a floating point number with an optional sign followed by `e` followed by an integer number which may be preceded by an optional sign. In case the floating point number is an integer, the dot may be omitted. Furthermore, if it is less than one, an initial zero may be omitted. Thus, `.3e-8` and `+42e0` are valid numbers in scientific notation.

(c) Derive *one* NFA that accepts $w \in L_{\text{token}}$, where token is either a comment or a number in scientific notation as defined in the previous tasks.

(d) Solve the simple matching problem on the NFA from above for the input string `-17.42e+23`.

(e) For a regular expression r and natural numbers n, m with $n \leq m$ we define a *repetition operator* $r^{[n,m]}$ to denote n to m occurrences of r. For instance, $L(a^{[1,3]}) = \{a, aa, aaa\}$. Prove or disprove that for each regular expression with repetition operators, there exists a regular expression without repetition operators.