TDT4205 Problem Set 1
Spring 2017

Answers are to be submitted via *It's Learning*, by Jan. 31st, 20:00.

1 Regular Languages

1.1

Variable declarations in Java can be preceded by zero or more of the modifier keywords `public, protected, private, static, abstract, final, native, synchronized, transient, volatile, strictfp`. A sequence of modifiers may not contain repetitions, or mutually exclusive combinations such as `public private`, but valid combinations can appear in any order. Is the set of valid modifier combinations a regular language? Justify your answer.

1.2

From "The C Programming Language" (Kernighan & Ritchie), pp. 194:
A floating constant consists of an integer part, a decimal point, a fraction part, an 'e' or 'E', an optionally signed integer exponent, and an optional type suffix, one of 'f ', 'F', 'l' or 'L'. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the 'e' and the exponent (not both) may be missing. The type is determined by the suffix; 'F' or 'f ' make it 'float', 'L' or 'l' makes it 'long double'; otherwise it is 'double'.
Draw a deterministic finite automaton which accepts floating point constants in this format.

2 A Simple Language of Stack Operations

These exercises will be concerned with a toy language that allows 32-bit integer arithmetic on a simple stack machine. Its environment contains an integer variable to implement a register, an array to implement a run time stack, and reacts to the tokens `INT, PUSH, POP, ADD, SUB, MUL, DIV, OUT, END`, as follows:

- INT converts the contents of the character buffer to a value, and stores it in the register
- PUSH adds the register value to the top of the stack
- POP removes the top value from the stack and stores it in the register
- ADD adds the contents of the register to the top value on stack
- SUB subtracts the contents of the register from the top value on stack
- MUL multiplies the top value on stack by the contents of the register
- DIV divides the top value on stack by the contents of the register
- OUT prints the register value on the standard output stream
- END stops execution

The provided archive `simplestack.tgz` implements these operations, but it is missing a text scanner.

Note

This exercise is not sufficiently specified to dictate a unique solution; its purpose is to highlight the technical choices which emerge when putting DFA simulation into practice. Make assumptions where necessary, state what they are, and freely modify the provided starting point as you see fit.

2.1

Design a DFA that recognizes the regular expressions `[0-9]+, PUSH, POP, ADD, SUB, MUL, DIV, OUT`.

2.2

Fill its transition table into a global array in the provided program, using the function `setup_table` (assume that keywords are separated by whitespace).

2.3

Implement DFA simulation in the function `next`, using your transition table so that the function returns one `token_t` value scanned from the `input` character stream on each invocation (or the `END` token when the character stream ends).

2.4

The register and stack of our system can handle negative integers, but the input instruction format does not. Suppose that we were translating a source language of expressions that include negative numbers, and therefore need to express negatives in terms of the supported operations.
Briefly describe a systematic translation method for negative terms.