TDT4205 Problem Set 1
Spring 2016

Answers are to be submitted by *It's Learning*, by Feb. 15^{th}, 20:00.
Please submit your answers as an archive *username*.tar.gz containing a PDF file with answers to theoretical questions, and a code directory like the provided one, containing all files required to build your solution. **Note that this deadline is due to the formal registration deadline for the course. It is highly recommended that you complete this exercise within a week, as further problem sets will be issued.**

1 Regular Languages

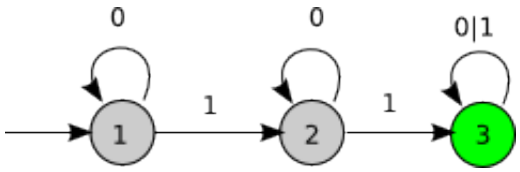

Figure 1: DFA (Initial state 1, accepting state 3)

1.1

Consider the set of binary strings Σ^* when $\Sigma = \{0, 1\}$. Which of these strings are **not** accepted by the DFA in Fig. 1?
(A verbal description suffices to answer this question, formal notation is not required.)

1.2

Write a regular expression for language of complex numbers in (finite) decimal notation, e.g. 0, 127.5, $2.7182 - 3.1425i$, $0 + 3i$, etc. (Assume that any pure imaginary number will still be written with the real part, i.e. with prefix 0+ or 0-, and use character classes, positive closure and zero-or-one-instance notation as in Dragon 3.3.5.)

1.3

Is the notation for regular expressions (Dragon 3.3.3) itself a regular langugage? Justify your answer.

2 A Simple Language for Drawing Lines

These exercises will be concerned with a minimal toy language to create simple line drawings on a page. The context is that of an imaginary pencil point initially located near the top left corner of the page. It is controlled by the three commands turn, draw, move, and end:

- Draw draws a line of fixed length at a given angle (initially, straight towards the right), leaving the pencil point at its end.
- Turn alters the angle of the next step by 30 clockwise.
- Move shifts the point by one step without drawing the line in between.
- End terminates the program.

The three operations are already implemented as the functions `turn()`, `draw()` and `move()`; what remains to be implemented is an automaton which recognizes the corresponding keywords from a text stream, and calls the appropriate function. The three keywords are to be case-insensitive (i.e. turn, TuRn and TURN are all correct). Characters not part of a command are to be ignored, such that text, whitespace and commands may be freely mixed in the input.

Note: this specification is somewhat open to interpretation - if you find some ambiguity, select an interpretation and state the reason for your choice.

2.1

Draw a deterministic finite automaton which recognizes the three specified commands and consumes/ignores other input until a word has been recognized.

2.2

The provided archive *pencil.tgz* contains code to translate the specified language into a postscript graphics file, except that it is missing a scanner. Implement your DFA from the previous task in `scanner.c`, to complete the program.