41080 Theory of Computing Science
Assignment 1
Finite Automata

1 Assessment Item Details

Due Date:	11:59pm, Friday 6th September
Type:	Programming Task & Report
Length:	Program – Approx. 500 Lines of Code
	Report – 2-3 pages
Submission:	Electronic through UTSOnline
Weighting:	30%
Type:	Individual

2 Preamble

Finite Automata are one of the simplest useful models of computation. Although they are limited in power, many of the simpler tasks that we automate with computers can be thought of as the action of a finite automata (or more strictly a finite state transducer, as we normally expect output beyond "accept" or "reject"). Event driven systems of all kinds (for example the event loop of a GUI application) are typically expressable as finite automata, and often explicitly are. Many of the subcomponents of digital eletronics can also be thought of as hardware implementations of finite automata. Even a cursory web search turns up applications in all sorts of interesting places.

Of particular interest for this assignment, the task of lexical analysis (whether it be in the traditional compiler design setting, or for data processings, natural language processing, artificial intelligence, &c.) is typically expressed via a finite automata.

To help introduce you to a basic model of computation, its applications and how we might go about translation theory into practice, this assignment focuses on implementing a simple lexical analyser for a basic calculator using a finite automaton based approach.

3 Your Task

This assignment consists in two components, a program and a report.

3.1 The Program (Weighting: 70%)

You will implement a [deterministic] finite automaton/transducer that takes in a string representing an arthimetic statement, and produces an output sequence of tokens that represent the numbers and operators.

Your program should also handle two types of errors – incorrect representation of a number, and incorrect representation of an expression.

Numbers will be in the following format:

- A string of digits possibly followed by a decimal point and a non-empty string of digits.
- If there is a decimal point, the string before the decimal point should consist only of "0".
- If the number starts with "0", it is either just "0" or "0.[something]" where the [something] is a string of digits.

The following expressions are valid:

- A valid number is a valid expression.

- Any valid expression, followed by an operator, followed by a valid number.
- The operators and numbers may or may not be separated by white space.

The program will be implemented in Java, unless specifically negotiated otherwise with a good reason (the subject coordinator reserves final judgement on what constitutes a good reason). Implementation in other languages will also incur extra work (constructing tests to demonstrate correctness).

3.2 The Report (Weighting: 30%)

The report is to be a simple report recording how your program utilises the material from the subject. In particular, you may wish to indicate some of the following:

- How did you extract finite automata from the specification?
- What is the formal specification of your automata?
- How does your code implement the finite automata that you designed?
- Did you use any additional techniques to improve the automata?
- Did you encounter any challenges or limitations, either technical or conceptual, in implementing a theoretical construct in a concrete programming language?

4 Marking

4.1 The Program

The marking for the program will be based on automated unit tests, examples of which will be provided. Tests (unless by specific arrangement) will use JUnit, and you are encouraged to write your own tests (which you can also report on). The marks for the program have one significant caveat: if the program does not use an automata based approach, it will be deemed to have failed to fulfil the specfication, regardless of functionality, and you will receive 0 for the program.

4.2 The Report

The report will be marked over three components: demonstration and understanding of subject material, accurate reportage of the content of the submitted code, clarity and precision of writing.

These will be marked qualitatively according to the following rubric in Table 1. Note that the marking criteria are considered to be cumulative; *e.g.* to achieve a credit, you must first meet the criteria for a pass.

Table 1: Marking criteria rubric for each component for each grade.

Component	Weighting	Pass	Credit	Distinction	High Distinction
Demonstration and understanding of subject material	15%	The report refers to the basic subject material related to the assignment.	The report correctly employs basic subject material to describe the program.	The report demonstrates the use of subject material in the development and design of the program.	The report demonstrates the use of techniques introduced in the subject to validate or otherwise improve or justify the design.
Accurate reportage of the submitted code	5%	The report makes no false claims about the code.	The report makes true claims about the code in relation to the subject material.	The report accurately demonstrates how the subject material is used in the code.	The report identifies the influence of the implementation conditions on how the subject material is employed.
Clarity and precision of writing	10%	The report has no major spelling or grammar errors.	The report is clearly and logically formatted.	The report requires no special effort to identify the concepts and code segments it is referring to.	The report could be used as a learning tool in this subject.