Theory of Computing

Prac 1

Name :	Student Number:
Mark awarded : /40	Demonstrator
I declare that this assignment is entirely my own work:	

1 Lexical Analyser

Most modern compilers use simple finite state machines to recognise variables, numbers, brackets, and operators. This question looks at recognising different types of numbers efficiently. Some examples of different numbers include:

- `1.02f` - float.
- `1.023` - double.
- `10008` - integer.
- `1003L` - long.

Fortunately page 15 of your notes has a non-deterministic finite state machine which can recognise the different valid types, unfortunately non-deterministic machines are inefficient and messy to implement.

(a) (5 marks) Convert the automaton into a *deterministic* finite-state machine. Hand in your new design as part of your solution (Pencil and paper is fine, as long as it's neat).

(b) (5 marks) Implement your design in Java. Your solution should be able to read in a file of numbers and print out the correct type of each number. If a number is invalid your solution must also print this out. There is some test data available on the course website to help you get started.

2 Fast String Matching

In Bioinformatics researchers often need to search long sequences of DNA for specific sub-sequences. DNA has 4 bases:

1. `A` - Adenin.
2. `C` - Cytosine.
3. `G` - Guanine.
4. `T` - Thymine.

There is a large file of hypothetical DNA data available off the course website. Using this data we wish to investigate the most efficient means of finding the following DNA sequence:

`TTTTTTTTTTTTCGTTGCCACTTAGCTTTGTGTTTTTG`
(also available off the course website to prevent spelling errors.)

(a) (4 marks) Implement the naive method of finding matching subsequences using two nested *for* loops. Since there is a large amount of repetition in the data this method should perform quite poorly.

(b) (8 marks) Write a program which constructs[1] a finite-state machine that recognises when the search string has been read. By using a finite-state machine we only have to examine every character once. The heart of the finite state machine will be a table of the following form:

	S_0	S_1	S_2	S_3	S_4	...
A	0	0	0	0	0	...
C	0	0	0	0	0	...
G	0	0	0	0	0	...
T	1	2	3	4	5	...

After every character has been read, the next state to go to will be found in the current state's column and the current letter's row. State S_i represents the state of having succesfully matched i characters. If the final state is reached then the entire search string has been matched.

(c) (3 marks) Time both your methods using `System.currentTimeMillis();` How do the methods compare?

3 Assorted problems

Solve each task below, using either a finite-state machine, a pushdown automata or a Turing machine. To show that you understand the limitations of each architecture it is important that you use the *least* powerful architecture for each problem.

(a) (5 marks) Only strings which contain twice as many 'a's as 'b's are accepted.

(b) (5 marks) Only decimal numbers which are divisible by 3 are accepted[2].

(c) (5 marks) Only statements which correctly describe the *min* operation are accepted. The numbers can be expressed in *unary* for simplicity and can be formatted as you see fit (as long as the numbers remain distinct from each other). Provide a short written description of how the machine works.

$$\begin{aligned} \min(1, 11) &= 1 \\ \min(11, 1) &= 1 \\ \min(111111, 111) &= 111 \\ &\dots \end{aligned}$$

[1] Your solution must include a *general* algorithm for creating the finite-state machine. It is no good to create this table by hand.

[2] Hint: consider the sum of the digits.