Assignment 2

Deadline:	Hand in by 5pm on Friday 3rd June 2016 (End of Semester 1)
Evaluation:	15 marks – which is 15% of your final grade
Late Submission:	1 Mark off per day late
Work:	This assignment is to be done **individually** – your submission may be checked for plagiarism against other assignments and against Internet repositories. If you adapt material from the Internet acknowledge your source.
Purpose:	To reinforce ideas covered in the lecture for understanding Turing machines and computation.

Description:

A Turing machine *M* can be defined by

$$M = (Q, \Sigma, I, q_0, \delta, F)$$

where:

Q – is the **states** of *M* (a finite set)
Σ – is the **alphabet** of *M* (a finite set with at least two symbols)
I – is the **input alphabet** of *M* (a non-empty subset of Σ)
q_0 – is the **initial state** of *M* (a member of Q)
δ – is the **instruction table** of *M*
F – is the **final states** of *M*

Write an implementation of a one-way infinite Turing Machine simulator in the programming language of your choice. This implementation must load in a Turing Machine description from a file (the format of these files is defined below), initialise the machine appropriately and then simulate the machine. Your implementation of the Turing Machine should (as far as is possible with limited memory) implement an infinite list to represent the tape; it should not be limited to an arbitrary size.

The first non-comment line will define all of the possible **states** Q of the machine separated by spaces. The states are represented by a string of characters (but must not include spaces or the hash symbol). The next line will define the **alphabet** Σ, again separated by spaces and each represented by a string of characters. Then the input to the machine (the initial symbols on the tape) is defined. The **input alphabet** I is not specifically defined but must be a subset of Σ. The next line should define the **initial state** of the machine q_0. Next the **final states** F of the machine are defined; if the machine reaches one of these states then it should finish execution. Finally the **instruction table** δ is defined one instruction at a time (each on a separate line). These instructions should have the form: `state symbol newState newSymbol movement`

Blank symbol - the symbol consisting of a single character ^ is reserved to represent a blank symbol. If a square on the tape to the right of the initial word is accessed, it should be initialised to ^.

Halting - if the machine tries to move past the first square of the tape (in the left direction) the machine should halt and fail. It should also fail if the instruction table δ does not contain an appropriate instruction for the combination of the current state and symbol.

You **must** follow the next two specifications in **each and every assignment for this course:**

1. Place the following comments at the top of your program code and **provide the appropriate information**:

```
/* Family Name, Given Name, Student ID, Assignment number, 159.331 */
/* explain what the program is doing . . . */
```

2. Ensure that your program uses print to send this information to the console. You might use:

```
print "--------------------------------------"
print " 159.331 Assignment 2 Semester 1 2016 "
print " Submitted by: Mouse, Mickey, 12345678 "
print "--------------------------------------"
```

Hand-in: Submit **your program and documentation (a zip file is acceptable)** electronically through the form on the stream site.
Marks will be allocated for: correctness, fitness of purpose, sensible use of data structures and algorithms, utility, style, use of sensible **comments and program documentation**, and general elegance. Good comments will help me to award you marks even if your code is not quite perfect.

If you have any questions about this assignment, please ask the lecturer.

Example:
An example program is shown below:

```
# Binary Increment (binary_increment.tm)
# Increments the value of a binary number by 1
# States - Q
0 1 2 H
# Alphabet - Sigma
0 1 ^
# Input Word - word of I
^ 1 1 0 1 1 0 1 0 1
# Initial State - q0
0
# Final States - F
H
# Instruction Table - Delta
#
0 ^ 1 ^ +1
0 0 0 0 -1
0 1 0 1 -1
1 ^ 2 1 -1
1 0 2 1 +1
1 1 1 0 +1
2 ^ H ^ +1
2 0 2 0 -1
2 1 2 1 -1
```

When a machine is run, the program should print out the values of the tape after each step of the computation (mark the square where the head is positioned). Only print the part of the tape that is actually used (squares that were either part of the input or have ever been accessed). You may want to print an extra blank cell on each side in case the head moves past these values. An example of the above program running is shown on the next page.

```
States (Q): 0 1 2 H
Symbols (S): 0 1 ^
Input Word (I): ^ 1 1 0 1 1 0 1 0 1
Initial State (q0): 0
Final States (F): H
Instruction Table (delta):
0 0 -> 0 0 -1
0 1 -> 0 1 -1
0 ^ -> 1 ^ 1
1 0 -> 2 1 1
1 1 -> 1 0 1
1 ^ -> 2 1 -1
2 0 -> 2 0 -1
2 1 -> 2 1 -1
2 ^ -> H ^ 1
Simulating Machine
0 : ^<1 1 0 1 1 0 1 0 1
1 : ^ 1<1 0 1 1 0 1 0 1
1 : ^ 0 1<0 1 1 0 1 0 1
1 : ^ 0 0 0<1 1 0 1 0 1
2 : ^ 0 0 1 1<1 0 1 0 1
2 : ^ 0 0 1<1 1 0 1 0 1
2 : ^ 0 0<1 1 1 0 1 0 1
2 : ^ 0<0 1 1 1 0 1 0 1
2 : ^<0 0 1 1 1 0 1 0 1
H : ^ 0<0 1 1 1 0 1 0 1
Program Halted (Success)
```

This program, along with some other test cases will be provided on the stream site.