A.1. The Turing machine simulator.

A.1.1. General description. A Turing machine simulator applet has been written by Tom Clarke for his final year project. It is available from the course web pages at `http://www.inf.ed.ac.uk/teaching/courses/ci`. When you first load the relevant page a certificate will be provided and you need to confirm that you trust it. The interface is self explanatory. You can upload Turing machines from files[1] (written in the description language given below) and then supply the initial tape input. After this you can start the simulator; you can set the speed at any point. Another useful feature is that you can step back to re-observe the effect of transitions.

The simulator also supports multi-track machines (the notation of Note 4 is used for tape symbols, so that $\langle a, b\rangle$ denotes a symbol where a is on the upper track and b on the lower track). The simulator will allow you to vary the number of tracks, since all they are is a convenient device for certain structured symbol names. As an extra feature the simulator issues a warning if there is a transition such as `(q1,<0,1>,q2,<1,0>,R)` that changes more than one track at the same time, since this goes against the idea of tracks. No warning is issued if the transition involves symbols with different numbers of tracks (say from 2 tracks to 3 tracks).

A.1.2. The Turing machine description language. The user must create a file that contains a description of the machine to be simulated. The description language has been designed with an eye to making the descriptions look as similar as possible to those that can be found in Note 3. Thus, for example, the palindrome recognizer, M_{palin}, of Note 3 might be presented to the simulator in the form of the following file:

```
Q = {q0, q1, q2, q3, q4, q5, q6}
I = q0
F = q6
G = {0, 1, b}
S = {0, 1}
D = {(q0,b,q6,b,R)*, (q0,0,q1,b,R)*, (q0,1,q3,b,R)*,
     (q1,b,q2,b,L)*, (q1,0,q1,0,R),  (q1,1,q1,1,R),
     (q2,b,q6,b,R)*, (q2,0,q5,b,L)*,
     (q3,b,q4,b,L)*, (q3,0,q3,0,R),  (q3,1,q3,1,R),
     (q4,b,q6,b,R)*, (q4,1,q5,b,L)*,
     (q5,b,q0,b,R)*, (q5,0,q5,0,L),  (q5,1,q5,1,L)}
```

[1] By default only files with the extension `.tm` are shown when browsing, however this can be changed.

(The significance of the * symbols will be discussed below.) It will be seen that the description consists of six *defining equations*. This seems somewhat at variance with the Turing machine defined in NOTE 3, which is a septuple. The discrepancy is explained by the fact that simulator treats the character `b` as the distinguished blank symbol; however the user can redefine the blank symbol to be some other (printable) character by means of an equation (`B = b` is an example, admittedly this changes nothing!). Note that the simulator always displays the blank character as an empty square on the tape. In general, objects must be defined before they are referred to. This imposes an ordering on certain pairs of definitions; thus the set of states (`Q`) must be defined before the transition function (`D`). There are no further restrictions on the ordering of definitions. The format of the input file has no significance except that any character not specified in the input alphabet will result in an error message. Missing braces etc. in the input file could possibly cause the program to hang (though it has been tested extensively and is in fact tolerant of some missing characters where no confusion can result), so please make sure that all the braces and commas are correct before starting the program.

A brief description of each of the six defining equations follows. Each definition is introduced by a key-letter, which indicates the thing being defined, followed by '=', an equals sign.

`Q:` The set of states is presented as a list of identifiers, separated by commas, and enclosed in braces, i.e., { and }. The identifiers are arbitrary sequences of alphanumeric characters. (Some special characters, such as { and } are not allowed.)

`I:` The initial state of the machine (the q_I of NOTE 3).

`F:` The final state of the machine (the q_F of NOTE 3).

`G:` The tape alphabet, Γ, of the machine. The tape symbols can be almost any printing characters. Exceptions include non-printable characters such as (space), and `?` (question mark); the latter has a special meaning to be explained shortly. The characters are separated by commas and enclosed in braces.

`S:` The input alphabet $\Sigma \subset \Gamma$.

`D:` The transition function is presented as a list of quintuples, separated by commas, and enclosed in braces. The components of each quintuple specify, in order, the old state, old tape symbol, new state, new tape symbol, and head movement (`L` for *left*, and `R` for *right*).

It is now necessary to explain the meaning of the wildcard symbol `?`, which can stand in the place of a tape symbol. Consider first the case when a wildcard symbol occurs as the second (old symbol) component of a tuple. Suppose that the transition function contains a tuple of the form $t = (q, \texttt{?}, \cdot, \cdot, \cdot)$; suppose also that the simulated machine is in state q, scanning the tape symbol s, and there is no tuple which has q and s as its first two components. In this situation, the tuple t acts as a default, specifying the next transition of the machine. Now consider the case of a wildcard symbol appearing as the fourth (new symbol) component of a tuple. In this position, the wildcard symbol always stands for the symbol which is currently scanned by the tape head. Effectively, when the machine is making a transition under the control of such a tuple, the tape contents of the machine remain unchanged. It is legitimate to include a tuple which has a wildcard character in the new symbol position but not in the old symbol position, but a moment's thought will reveal that there is no virtue in doing this.

Each tuple may optionally be followed by an asterisk. Tuples marked with an asterisk indicate *printing transitions*; when the simulated machine undergoes a printing transition, a record of the transition is recorded as part of the trace (see under the Tools menu) which can be saved to a file if desired. Except for computations which are expected to halt within a few tens of steps, asterisks should be used with discretion and sparingly, otherwise the file which records the computation will grow very long. (You can also select or modify which transitions cause printing by using Preferences under the Tools menu).

A.1.3. Pragmatics. A ubiquitous component in the construction of Turing machine programs is a routine which searches right or left along the tape for the first occurrence of a certain tape symbol s. This component can be implemented very economically using the wildcard feature. Assume that the search is to the right. One merely has to introduce a new state, q say, and a pair of tuples

$$(q, \texttt{?}, q, \texttt{?}, \texttt{R}) \quad \text{and} \quad (q, s, q', s', d),$$

where q', s', and d specify appropriate actions for the machine to take when encountering the symbol s.