Universal Turing Machine
(--- this time I mean it!)

Announcement
No class this coming Friday and Monday.
Updated lecture schedule posted on website.

Turing Machine Descriptions
A description of a TM is a string of 0's and 1's that expresses the TM in some standardized format, that is simple enough to be interpreted by another TM.

For concreteness, standardize on:

```
header | transitions
```

where header = $0^n 10^m 10^k 10^l 10^s 10^t 0^n 10^m$

- $n =$ # states
- $m =$ # working tape symbols
- $k =$ # input tape symbols (first $k$ out of $m$)
- $s,t,r =$ start, accept, reject states (in range 1...$n$)
- $u,v =$ blank and left endmarker symbols (1...$m$)
The transition rule $S$ is represented as a sequence of strings (arbitrary order) in the "transitions" block of code. Each of them is:

$$0^1 0^a 1^b 0^c 1^d 1$$

if the machine, when reading symbol $a$ in state $p$, writes $b$, transitions to $q$, moves in direction $d$ where:

$$d = \begin{cases} 1 & \text{for direction } -1 \\ 2 & \text{for direction } 0 \\ 3 & \text{for direction } +1 \end{cases}$$

A universal Turing machine is one that takes an input string $x \# y$ (where $x$, $y$ are both written in 0's + 1's) and:

- if $x$ is not the description of a TM, it rejects input $x \# y$.
- if $x$ is a description of TM $M$ and $y$ is not a valid input description, it rejects $x \# y$.
- if $x \# y$ represents a TM $M$ and input string $\xi$ then the UTM simulates $M$ processing input $\xi$ and accepts/rejects it if $M$ accepts/rejects $\xi$.
For a Turing machine with input alphabet of size $k$, an input description is a binary string $0^{a_1}10^{a_2}10^{a_3}1\ldots 0^{a_e}1$ representing the input string $a_1a_2\ldots a_e$ where $1 \leq a_1, \ldots, a_e \leq k$.

We can describe the contents of $M$'s working tape in binary similarly by using run-length encoding with runs of 0's of length ranging 1, ..., $m$. ($m$ = size of working alphabet.)

**Def.** The configuration of a TM is $(p, j, z)$ where

- $p \in Q$ is a current state,
- $j \in \mathbb{N}$ is the current read/write head position,
- $z$ is a natural number representing working tape contents up to and including last non-blank symbol, (not including left end marker).

Configurations can be run-length encoded as $0^{p}10^{j}10^{z_1}10^{z_2}1\ldots 10^{z_e}1$ where $e = \text{length of } z$. 
Skeleton for UTM: Multi-tape with

Tape 1 = input tape
Tape 2 = simulated working tape
Tape 3 = state tape

UTM:

// copy x from input tape to Tape 2
while (not reading # on Tape 1)
    \( \sigma \) = symbol on Tape 1
    write \( \sigma \) on Tape 2
    move right on Tapes 1, 2
endwhile // now Tape 2 contains x
if IsValidTM(Tape 2) returns false:
    Enter reject state, r
// clear Tape 2
while (not reading 1 on Tape 2)
    write blank symbol on Tape 2
    move left
endwhile
while (not reading \( \downarrow \) on Tape 1)
    \( \sigma \) = symbol on Tape 1
    write \( \sigma \) on Tape 2
    move right on Tapes 1, 2
while \( \text{now } y \text{ is on Tape 2} \)

if \( \text{IsValid Input (Tape 1, Tape 2)} \) return false:

Enter reject state, \( r \)

\( \# x y \) is a valid TM and its input

Now we want to actually simulate

\( x \) running on input \( y \)

**INITIAL CONFIG** (Tape 1, Tape 2)

\( \) Takes \( x \# y \) on Tape 1.

\( \) Write initial configuration of

\( \) \( x \) processing \( y \) on Tape 2.

\( \) \( \begin{array}{c} 0 s 1 0 1 y \\ \hline \text{start state } s, \text{symbols on tape} \end{array} \)

\( \) location \( 1 \)

repeat forever:

**SINGLE STEP** (Tape 1, Tape 2)

\( \) simulates one transition of \( M \)

\( \) descriptor of \( M \) resides on Tape 1

\( \) Tape 2 holds configuration of \( M \)

\( \) function overwrites Tape 2

\( \) with the config after one

\( \) transition.

if \( \text{Test Accept (Tape 1, Tape 2)} \)

Enter accept state, \( t \).
How to implement Single Step?

1. Test Register (Type 1, Type 2)
   - Enter register state in
   - Else break the loop and halt