How to traverse?

```
traverse([Token|R], Current, Final):-
        move(Current, Token, Next),
        traverse(R, Next, Final).

traverse([], Final, Final).
```

- Take item from head of list
- Make move to new state
- Call recursively until end of list

Turing Machine?

- How might you implement one?
- How might you represent the tape?
- Need to have some notion of actions associated with particular states:
  - Move tape
  - Mark tape

A way to represent the tape

- Lots of possible ways to do this
- Use a list
- Some notion of pointing into the list to a particular location
  - Not obviously easy to move back and forth using mechanisms available in Prolog (recursion?)
- I'm going to use two lists
  - “the bit of the tape on the left”
  - “the bit of tape on the right”

A way to represent the tape

- In my program I have defined operations to move left and right along the tape
  - Move items from one list to the other
- This means that my traverse predicate now has some extra arguments
  - The left and right parts of the tape
- My tape is preloaded with month names
  - But I can only search one way through it.

Details

- Underscore: “_” is a bit special
  - It matches anything
- My implementation presented here is dumb:
  - It only goes one way along the tape
  - It assumes all months have less than 30 days
  - It deals with years dumbly
- How else could it be dumb?

For the assignment...

- Take this code as a starting point if you wish...
- Take time to pull it apart and play with it
- Test it in various scenarios
- Then take some time to annotate it carefully and start to make a plan about how you want to make it do as you require
- Write down a bit of a plan before you start to hack away at Prolog