--- Day 20: Pulse Propagation ---

With your help, the Elves manage to find the right parts and fix all of the machines. Now, they just need to send the command to boot up the machines and get the sand flowing again.

The machines are far apart and wired together with long cables. The cables don't connect to the machines directly, but rather to communication modules attached to the machines that perform various initialization tasks and also act as communication relays.

Modules communicate using pulses. Each pulse is either a high pulse or a low pulse. When a module sends a pulse, it sends that type of pulse to each module in its list of destination modules.

There are several different types of modules:

Flip-flop modules (prefix `%`) are either on or off; they are initially off. If a flip-flop module receives a high pulse, it is ignored and nothing happens. However, if a flip-flop module receives a low pulse, it flips between on and off. If it was off, it turns on and sends a high pulse. If it was on, it turns off and sends a low pulse.

Conjunction modules (prefix `&`) remember the type of the most recent pulse received from each of their connected input modules; they initially default to remembering a low pulse for each input. When a pulse is received, the conjunction module first updates its memory for that input. Then, if it remembers high pulses for all inputs, it sends a low pulse; otherwise, it sends a high pulse.

There is a single broadcast module (named `broadcaster`). When it receives a pulse, it sends the same pulse to all of its destination modules.

Here at Desert Machine Headquarters, there is a module with a single button on it called, aptly, the button module. When you push the button, a single low pulse is sent directly to the `broadcaster` module.

After pushing the button, you must wait until all pulses have been delivered and fully handled before pushing it again. Never push the button if modules are still processing pulses.

Pulses are always processed in the order they are sent. So, if a pulse is sent to modules `a`, `b`, and `c`, and then module `a` processes its pulse and sends more pulses, the pulses sent to modules `b` and `c` would have to be handled first.

The module configuration (your puzzle input) lists each module. The name of the module is preceded by a symbol identifying its type, if any. The name is then followed by an arrow and a list of its destination modules. For example:

```
broadcaster -> a, b, c
%a -> b
%b -> c
%c -> inv
&inv -> a
```

In this module configuration, the broadcaster has three destination modules named `a`, `b`, and `c`. Each of these modules is a flip-flop module (as indicated by the `%` prefix). `a` outputs to `b` which outputs to `c` which outputs to another module named `inv`. `inv` is a conjunction module (as indicated by the `&` prefix) which, because it has only one input, acts like an inverter

(it sends the opposite of the pulse type it receives); it outputs to `a`.

By pushing the button once, the following pulses are sent:

```
button -low-> broadcaster
broadcaster -low-> a
broadcaster -low-> b
broadcaster -low-> c
a -high-> b
b -high-> c
c -high-> inv
inv -low-> a
a -low-> b
b -low-> c
c -low-> inv
inv -high-> a
```

After this sequence, the flip-flop modules all end up off, so pushing the button again repeats the same sequence.

Here's a more interesting example:

```
broadcaster -> a
%a -> inv, con
&inv -> b
%b -> con
&con -> output
```

This module configuration includes the `broadcaster`, two flip-flops (named `a` and `b`), a single-input conjunction module (`inv`), a multi-input conjunction module (`con`), and an untyped module named `output` (for testing purposes). The multi-input conjunction module `con` watches the two flip-flop modules and, if they're both on, sends a low pulse to the `output` module.

Here's what happens if you push the button once:

```
button -low-> broadcaster
broadcaster -low-> a
a -high-> inv
a -high-> con
inv -low-> b
con -high-> output
b -high-> con
con -low-> output
```

Both flip-flops turn on and a low pulse is sent to `output`! However, now that both flip-flops are on and `con` remembers a high pulse from each of its two inputs, pushing the button a second time does something different:

```
button -low-> broadcaster
broadcaster -low-> a
a -low-> inv
a -low-> con
inv -high-> b
con -high-> output
```

Flip-flop `a` turns off! Now, `con` remembers a low pulse from module `a`, and so it sends only a high pulse to `output`.

Push the button a third time:

```
button -low-> broadcaster
broadcaster -low-> a
a -high-> inv
a -high-> con
inv -low-> b
con -low-> output
b -low-> con
con -high-> output
```

This time, flip-flop `a` turns on, then flip-flop `b` turns off. However, before `b` can turn off, the pulse sent to `con` is handled first, so it briefly remembers all high pulses for its inputs and sends a low pulse to `output`. After that, flip-flop `b` turns off, which causes `con` to update its state and send a high pulse to `output`.

Finally, with `a` on and `b` off, push the button a fourth time:

```
button -low-> broadcaster
broadcaster -low-> a
a -low-> inv
a -low-> con
inv -high-> b
con -high-> output
```

This completes the cycle: `a` turns off, causing `con` to remember only low pulses and restoring all modules to their original states.

To get the cables warmed up, the Elves have pushed the button `1000` times. How many pulses got sent as a result (including the pulses sent by the button itself)?

In the first example, the same thing happens every time the button is pushed: `8` low pulses and `4` high pulses are sent. So, after pushing the button `1000` times, `8000` low pulses and `4000` high pulses are sent. Multiplying these together gives `32000000`.

In the second example, after pushing the button `1000` times, `4250` low pulses and `2750` high pulses are sent. Multiplying these together gives `11687500`.

Consult your module configuration; determine the number of low pulses and high pulses that would be sent after pushing the button `1000` times, waiting for all pulses to be fully handled after each push of the button. What do you get if you multiply the total number of low pulses sent by the total number of high pulses sent?

To begin, get your puzzle input.

Answer: [Submit]

You can also [Share] this puzzle.