**[Advent of Code](http://adventofcode.com/)**

Clive Ward 11\*

* [[About]](http://adventofcode.com/about)
* [[Stats]](http://adventofcode.com/stats)
* [[Leaderboard]](http://adventofcode.com/leaderboard)
* [[Settings]](http://adventofcode.com/settings)
* [[Log out]](http://adventofcode.com/auth/logout)

**--- Day 7: Some Assembly Required ---**

This year, Santa brought little Bobby Tables a set of wires and [bitwise logic gates](https://en.wikipedia.org/wiki/Bitwise_operation)! Unfortunately, little Bobby is a little under the recommended age range, and he needs help assembling the circuit.

Each wire has an identifier (some lowercase letters) and can carry a [16-bit](https://en.wikipedia.org/wiki/16-bit) signal (a number from 0 to 65535). A signal is provided to each wire by a gate, another wire, or some specific value. Each wire can only get a signal from one source, but can provide its signal to multiple destinations**. A gate provides no signal until all of its inputs have a signal.**

The included instructions booklet describes how to connect the parts together: x AND y -> z means to connect wires x and y to an AND gate, and then connect its output to wire z.

For example:

* 123 -> x means that the signal 123 is provided to wire x.
* x AND y -> z means that the [bitwise AND](https://en.wikipedia.org/wiki/Bitwise_operation" \l "AND) of wire x and wire y is provided to wire z.
* p LSHIFT 2 -> q means that the value from wire p is [left-shifted](https://en.wikipedia.org/wiki/Logical_shift) by 2 and then provided to wire q.
* NOT e -> f means that the [bitwise complement](https://en.wikipedia.org/wiki/Bitwise_operation" \l "NOT) of the value from wire e is provided to wire f.

Other possible gates include OR ([bitwise OR](https://en.wikipedia.org/wiki/Bitwise_operation" \l "OR)) and RSHIFT ([right-shift](https://en.wikipedia.org/wiki/Logical_shift)). If, for some reason, you'd like to *emulate* the circuit instead, almost all programming languages (for example, [C](https://en.wikipedia.org/wiki/Bitwise_operations_in_C), [JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Bitwise_Operators), or [Python](https://wiki.python.org/moin/BitwiseOperators)) provide operators for these gates.

For example, here is a simple circuit:

123 -> x

456 -> y

x AND y -> d

x OR y -> e

x LSHIFT 2 -> f

y RSHIFT 2 -> g

NOT x -> h

NOT y -> i

After it is run, these are the signals on the wires:

d: 72

e: 507

f: 492

g: 114

h: 65412

i: 65079

x: 123

y: 456

In little Bobby's kit's instructions booklet (provided as your puzzle input), what signal is ultimately provided to *wire a*?

To begin, [get your puzzle input](http://adventofcode.com/day/7/input" \t "_blank).

Top of Form

Answer: 

Bottom of Form

Top of Form

Bottom of Form

You can also [Shareon [Twitter](https://twitter.com/intent/tweet?text=%22Some+Assembly+Required%22+%2D+Day+7+%2D+Advent+of+Code&url=http%3A%2F%2Fadventofcode%2Ecom%2Fday%2F7&related=ericwastl&hashtags=AdventOfCode" \t "_blank) [Google+](https://plus.google.com/share?url=http%3A%2F%2Fadventofcode%2Ecom%2Fday%2F7" \t "_blank) [Reddit](http://www.reddit.com/submit?url=http%3A%2F%2Fadventofcode%2Ecom%2Fday%2F7&title=%22Some+Assembly+Required%22+%2D+Day+7+%2D+Advent+of+Code" \t "_blank)] this puzzle.

Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> bin(128)

'0b10000000'

>>> bin(129)

'0b10000001'

>>> bin(256)

'0b100000000'

>>> bin(8) AND bin(1)

SyntaxError: invalid syntax

>>> bin(8) & bin(1)

Traceback (most recent call last):

File "<pyshell#4>", line 1, in <module>

bin(8) & bin(1)

TypeError: unsupported operand type(s) for &: 'str' and 'str'

>>> 5 & 1

1

>>> 5 or 1

5

>>> bin(5 & 1)

'0b1'

>>> 256 & 1

0

>>> 256 | 1

257

>>> 512 | 1

513

>>> bin(512 | 1)

'0b1000000001'

>>> a = bim(129)

Traceback (most recent call last):

File "<pyshell#12>", line 1, in <module>

a = bim(129)

NameError: name 'bim' is not defined

>>> a -

SyntaxError: invalid syntax

>>>

>>>

>>>

>>> a = bin(129)

>>> print a[1]

SyntaxError: Missing parentheses in call to 'print'

>>> print(a[1])

b

>>>