

```
--- Day 14: Docking Data ---
```

As your ferry approaches the sea port, the captain asks for your help again. The computer system that runs this port isn't compatible with the docking program on the ferry, so the docking parameters aren't being correctly initialized in the docking program's memory.

After a brief inspection, you discover that the sea port's computer system uses a strange **bitmask** system in its initialization program. Although you don't have the correct decoder chip handy, you can emulate it in software!

The initialization program (your puzzle input) can either update the bitmask or write a value to memory. Values and memory addresses are both 36-bit unsigned integers. For example, ignoring bitmasks for a moment, a line like `mem[8] = 11` would write the value `11` to memory address `8`.

The bitmask is always given as a string of 36 bits, written with the most significant bit (representing 2^{35}) on the left and the least significant bit (2^0 , that is, the 1s bit) on the right. The current bitmask is applied to values immediately before they are written to memory: a 0 or 1 overwrites the corresponding bit in the value, while an X leaves the bit in the value unchanged.

For example, consider the following program:

```
mask = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX0X  
mem[8] = 11  
mem[7] = 101  
mem[8] = 0
```

[illegible][illegible]

```
value: 00000000000000000000000000000000 (decimal 0)
mask:  XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX0X
result: 000000000000000000000000000000001000000 (decimal 64)
```

To initialize your ferry's docking program, you need the sum of all values left in memory after the initialization program completes. (The entire 36

Our **sponsors** help
make Advent of
Code possible:

TwilioQuest -
Learn to code and
lead your
intrepid crew on
a mission to save
The Cloud in
TwilioQuest, a PC
role-playing game
inspired by
classics of the
16-bit era. Free
forever, and
available now for
Windows, Mac, and
Linux.

bit address space begins initialized to the value 0 at every address.) In the above example, only two values in memory are not zero - 101 (at address 7) and 64 (at address 8) - producing a sum of 165.

Execute the initialization program. What is the sum of all values left in memory after it completes?

Your puzzle answer was 4297467072083.

--- Part Two ---

For some reason, the sea port's computer system still can't communicate with your ferry's docking program. It must be using version 2 of the decoder chip!

A version 2 decoder chip doesn't modify the values being written at all. Instead, it acts as a **memory address decoder**. Immediately before a value is written to memory, each bit in the bitmask modifies the corresponding bit of the destination memory address in the following way:

- If the bitmask bit is 0, the corresponding memory address bit is unchanged.
- If the bitmask bit is 1, the corresponding memory address bit is overwritten with 1.
- If the bitmask bit is X, the corresponding memory address bit is floating.

A floating bit is not connected to anything and instead fluctuates unpredictably. In practice, this means the floating bits will take on all possible values, potentially causing many memory addresses to be written all at once!

For example, consider the following program:

```
mask = 00000000000000000000000000000000X1001X
mem[42] = 100
mask = 00000000000000000000000000000000X0XX
mem[26] = 1
```

When this program goes to write to memory address 42, it first applies the bitmask:

```
address: 00000000000000000000000000000000101010 (decimal 42)
mask:    00000000000000000000000000000000X1001X
result:  00000000000000000000000000000000X1101X
```

After applying the mask, four bits are overwritten, three of which are different, and two of which are floating. Floating bits take on every possible combination of values; with two floating bits, four actual memory addresses are written:

```
0000000000000000000000000000000011010 (decimal 26)
0000000000000000000000000000000011011 (decimal 27)
00000000000000000000000000000000111010 (decimal 58)
00000000000000000000000000000000111011 (decimal 59)
```

Next, the program is about to write to memory address 26 with a different bitmask:

```
address: 0000000000000000000000000000000011010 (decimal 26)
mask:    00000000000000000000000000000000X0XX
result:  000000000000000000000000000000001X0XX
```

This results in an address with three floating bits, causing writes to eight memory addresses:

[illegible]

The entire 36-bit address space still begins initialized to the value 0 at every address, and you still need the sum of all values left in memory at the end of the program. In this example, the sum is 208.

Execute the initialization program using an emulator for a version 2 decoder chip. What is the sum of all values left in memory after it completes?

Your puzzle answer was 5030603328768.

Both parts of this puzzle are complete! They provide two gold stars: **

You can also [\[Share\]](#) this puzzle.