CMPUT 274/5 - Tangible Computing **Winter 2020**
Morning Problem: Chasing Pointers

Description

Last week, I learned about pointers and memory for the first time. Unfortunately, I am finding it very confusing. Sometimes, a pointer is just an address to a value in memory. Other times, a pointer is the address of another pointer! I just can't wrap my head around all these pointers to pointers to pointers.

To try to understand pointers, I have been practicing by drawing out a region of memory on paper, and filling each position with either a value or a pointer to another memory address. Then, I take the address of the first element in memory and follow it to see where it leads. I do this by dereferencing the pointer as many times as needed to reach a value in the array.

However, I am running into a horrible problem. Sometimes, when I chase pointers this way I end up at an address I have already seen. Then I end up in an endless loop going to the same addresses over and over again!

Can you help me write a program to chase down these tricky pointers and find out which chases cause cycles?

Input

The first line of input will contain two space-separated integers: $0 \leq S < 2^{31}$, $1 \leq A \leq 1000$ where:

- S: the starting address of the allocated region of memory
- A: the size of (number of elements in) the allocated region in memory

You are also guaranteed that $S + A \leq 2^{31}$.

On the second line will be A space-separated integers, each lying in the range $[-2^{31}, 2^{31})$, indicating what is stored at address $S + i$. This will be either:

(a) A **value**, a strictly negative integer, or

(b) A **pointer (memory address)**, a nonnegative integer.

It is guaranteed that no chase will result in a memory access that is outside the bounds of the array.

Output

Output a single line, the result of "chasing" when starting with a pointer to address S. This will be one of the following two statements:

1. The negative value v found while chasing.
2. The message `There was a cycle`, if, when chasing down a pointer, the program revisits any address it has previously seen.

Comment

This fictitious machine is a bit strange. The memory addresses require 32 bits yet each pointer uses only one byte. An alternative view of this problem is that you are chasing indices into a strange array whose first index is S rather than the traditional start index of 0. But don't overthink this too much, just solve the problem as it is without worry about whether this could be realized by a "real" computer.

Sample Input 1

```
10428394 1
-12
```

Sample Output 1

```
-12
```

Explanation: There is only one element in this allocated region of memory. A chase at the starting address, 10428394 returns the value -12 imediately.

Sample Input 2

```
8000 6
8004 8003 -10 8002 -1 9000
```

Sample Output 2

```
-1
```

Explanation: The starting address of the allocated region is 8000. This means that valid addresses into the 6-element array range from 8000 to 8005 (inclusive). Following the first address, we find address 8004. At position 8004 is the value -1, which is printed.

Sample Input 3

```
4000 3
4001 4002 4000
```

Sample Output 3

```
There was a cycle
```

Explanation: Starting at position 4000, we dereference to get the value 4001. This is another valid address in the array, so if we dereference again, we get 4002, and then 4000 again, which leads back to 4001. This is an endless cycle.

Sample Input 4

```
0 7
3 -7 4 2 6 -10 4
```

Sample Output 4

```
There was a cycle
```

Explanation: Beginning a chase at address 0 leads to the following sequence: 0, 3, 2, 4, 6, 4. The chase would continue to cycle between 4 and 6 forever.