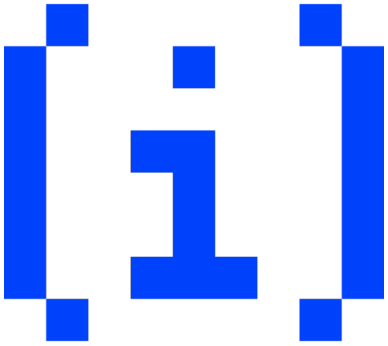


en[i]gma 0x02 Senior

December 15, 2024

Problems

Problem	Time Limit	Memory Limit	Points
The Treasure Key	1 sec	16MB	100
Christmas Treats	1 sec	16MB	100
The Magic Number	1 sec	64MB	100
The Old Typewriter	1 sec	64MB	100
Now You Think with Portals	1 sec	64MB	100
Total			500

The Treasure Key

Annoula and her brother Totos, while searching in the attic of their house for some old photo albums, discovered a forgotten chest. Inside the chest, there is an old map that once belonged to the legendary pirate Henry Every. According to the map, a precious treasure is hidden somewhere on the island, but to reach it, they must solve three puzzles.

Annoula and Totos are curious to see what the chest contains. However, they must first open the lock.

Puzzle 1

At the bottom of the chest, there are two numbers engraved. Next to these numbers, the pirate had also engraved a note:
«**1.** My crew consists of as many sailors as the product of the two numbers engraved here.
If you manage to find the product and enter the number in the lock, the chest will open!»

Puzzle 2

A little further down, it says:
«**2.** Once you open the chest, you only need to find where the treasure is hidden on the map.
From the numbers A and B, either A, B, or both A and B might be 0.
However, it's possible that neither of them is 0.

- If neither is 0, the treasure is hidden under the number 0 on the map,
- If only A is 0, the treasure is hidden under the number 1 on the map,
- If only B is 0, the treasure is hidden under the number 2 on the map,
- If both A and B are 0, the treasure is hidden under the number 3 on the map»

Totos and Annoula need your help because they're quite stressed and need to verify that they're doing the right calculations to eventually find the treasure!

We want to write a program that reads from input `STDIN` two integers, A and B (both positive, single-digit numbers), separated by a newline character (`'\n'`) and prints two lines in the output `STDOUT` δύο γραμμές.
In the first line, the program should print the product M of A and B.
In the second line, the program should print one of the numbers 0, 1, 2, or 3 based on the pirate's note **2.**

Examples

1st

Input (STDIN)

```
6
8
```

Output (STDOUT)

```
48
0
```

Explanation of the first example:

The crew of the captain consists of $6 \times 8 = 48$ sailors, and **neither** of the two numbers is 0, so the output is **0**.

2nd

Input (STDIN)

```
0
3
```

Output (STDOUT)

```
0
1
```

Explanation of the second example:

The crew of the captain consists of $0 \times 3 = 0$ 0 sailors, and the first number, A, is 0, so the output is **1**.

Note! Each line of input and output (should) end with a newline character `'\n'`.

A Little Cheat Sheet!

Python

```python
from sys import exit

# The execution of the program starts here!
if __name__ == "__main__":
    x = int(input())

    if(x == 1) : ...
    elif(x == 2) : ...

    exit(0)
```

C/C++ (with `cstdio`):

```c
#include <stdio.h>

# The execution of the program starts here!
int main() {
    int x;
    scanf("%d", &x);

    if(x == 1){...};
    else if(x == 2){...};

    return 0;
}
```

C/C++ (with `iostream`):

```c
#include <iostream>

# The execution of the program starts here!
int main() {
    int x;
    std::cin >> x; // Use std::cin for input

    if (x == 1) {
        // Add your code for x == 1 here
    } else if (x == 2) {
        // Add your code for x == 2 here
    }

    return 0;
}
```

Christmas Treats

In preparation for celebrating Christmas, Toto's and Annoula's mom made a bunch of sweets: melomakarona (1), kourabiedes (2), koulourakia (3), diples (4), and chocolates (5).

After completing them, she packed them in large boxes to give as gifts to the guests. Before starting to decorate the gift boxes, she realized she hadn't noted how many pieces of a particular sweet were in each box. This was quite important because each guest had a different favorite sweet, and each was assigned a different gift box. Immediately, Annoula and Totos sprang into action and started counting how many pieces of a particular type of sweet each box contained..

We want to write a program that will read two positive integers, N and S, from the input `STDIN`, separated by a space (`' '`), followed by S positive integers S_i, separated by a space (`' '`). The program will print the count of the occurrences of N among the S numbers.

- N represents the sweet for which we need to count how many times it appears in the box.
- S represents the total number of sweets in the box.
- S_i represents each sweet inside the box.

Note! Each input or output line must end with a newline character `'\n'`.

Example

Input (STDIN)

```
1 10
2 1 1 3 1 5 1 2 1 1
```

Output (STDOUT)

```
6
```

Explanation of the Example:

Totos and Annoula are looking to see how many pieces of sweet 1 there are among the 10 sweets in the box. Counting, they find that there are 6 such sweets.

The Magic Number

Totos is preparing to guess the magic number and win in the lucky game with the same name. To do this, he decided to create the following program on his computer to practice..

Initially, the program will read the magic number from the **input**. Then, it will read some numbers, as many as needed, until it finds one that matches the magic number. For each number it reads, it will print to the **output**:

- `1`, if the number read is smaller than the magic number,
- `2`, if the number read is greater than the magic number,
- `0`, when the number read equals the magic number.

Examples

1st

Input

```
100
5
100
```

Output

```
1
0
```

Explanation of the first example:

The input contains the magic number `100` and 2 guesses, `5` and `100`.
The output first prints `1` because $5 < 100$, and then prints `0` because $100 = 100$.

2nd

Input

```
10
11
9
12
10
```

Output

```
2
1
2
0
```

Explanation of the second example:

The input contains the magic number `10` and 4 guesses: `11`, `9`, `12` and `10`.
The output first prints `2` because $11 > 10$, then `1` because $9 < 10$, followed by `2` because $12 > 10$, and finally `0` because $10 = 10$.

3rd

Input

```
10
3
10
11
```

Output

```
1
0
```

Explanation of Example 3:

The input contains the magic number `10` and 3 guesses: `3`, `10` and `11`.
The output first prints `1` because $5 < 100$, and then prints `0` because $10 = 10$.
No further numbers are printed because the program **stopped** when the magic number was found.

The Old Typewriter

At Christmas, all the families gather for food and many, many, stories; old accomplishments of the elders, tricks of the young, and the antics of Koula, the neighbor. Isn't it a shame if these aren't recorded?

This is how Totos started secretly writing down the stories he hears. But he didn't want to take out a laptop next to the turkey, so he used a simple notebook. Later, Annoula asked him to type the stories out on their grandfather's old typewriter!

The typewriter uses a paper of special size that fits up to K $(20 \leq K \leq 200)$ characters per line (we assume all characters, including spaces, have the same size). Unfortunately, their grandfather had broken the hyphen (-) key, so you cannot break a word into two parts. Therefore, Totos needs to reformat the notes, changing where to break the text into lines, so that all lines fit within the paper size of the typewriter.

We want to write a program that will read from `STDIN` the integer K, the maximum line size of the paper, followed by lines containing words. Each word has a length **of at most 20** characters. The program will print to `STDOUT` each word from the input with appropriate line breaks so that each line contains at most K characters.
If a line contains fewer than K characters, the text should be left-aligned, as shown in the example.
Attention: The first line should begin with **4 spaces** (`'  '`), as it marks the beginning of a paragraph.

Subtasks

- The implementation of the above description will be graded for **70%** of the points.
- For an additional **20%** of the points, you need to structure the text to have separate paragraphs.
  When reading the word `PAR`, you should **not print it**, and instead leave a **blank line**.
  Κάθε παράγραφος (συμπεριλαμβανομένης και της πρώτης) θα πρέπει να ξεκινάει με **4 κενά**.
- For an additional **10%** of the points, the input may contain a word "split" into 2 lines using a hyphen ς (`-`); you should print these words together in the output.

Example

Input (STDIN)

```
42
GENTLE READER: This is a handbook about TeX, a new typesetting system
intended for the creation of beautiful books—and especially for books
that contain a lot of mathematics. By preparing a manuscript in TeX format,
you will be telling a computer exactly how the manuscript is to be transformed
into pages whose typographic quality is comparable to that of the world's finest
printers; yet you won't need to do much more work than would be involved if
you were simply typing the manuscript on an ordinary typewriter. In fact, your
total work will probably be significantly less, if you consider the time it ordinarily
takes to revise a typewritten manuscript, since computer text files are so easy
to change and to reprocess. (If such claims sound too good to be true, keep in
mind that they were made by TeX's designer, on a day when TeX happened to
be working, so the statements may be biased; but read on anyway.)
```

Output (STDOUT)

```
    GENTLE READER: This is a handbook
about TeX, a new typesetting system
intended for the creation of beautiful
books—and especially for books that
contain a lot of mathematics. By preparing
a manuscript in TeX format, you will be
telling a computer exactly how the
manuscript is to be transformed into pages
whose typographic quality is comparable to
that of the world's finest printers; yet
you won't need to do much more work than
would be involved if you were simply
typing the manuscript on an ordinary
typewriter. In fact, your total work will
probably be significantly less, if you
consider the time it ordinarily takes to
revise a typewritten manuscript, since
computer text files are so easy to change
and to reprocess. (If such claims sound
too good to be true, keep in mind that
they were made by TeX's designer, on a
day when TeX happened to be working, so
the statements may be biased; but read on
anyway.)
```

A Small Cheat Sheet!

Reading **an indefinite number of words** from input: To read the words following the number K from `STDIN`, you can use the following code:

Python:

```python
import sys

for word in sys.stdin.split():
^^I# The variable word stores 1 new word from the input
```

C/C++ (with `cstdio`):

```c
#include <stdio.h>
...
char word[21]; // maximum word length + 1
while (scanf("%s", word) == 1) {
^^I// The variable word stores 1 new word from the input
```

```
}
```

C++ (with `iostream`):

```
#include <iostream>
...
string word;
while (cin >> word) {
^^I// The variable word stores 1 new word from the input
}
```

To print in Python without changing the line immediately after, use the `end=''` parameter in `print()`, like this:

```
print(..., end='')
```

Now You Think with Portals

We are in a skyscraper and want to reach the T-th floor. We start from the S-th floor and can use 2 methods to change floors:

- via **ladders**: each ladder moves us from the current floor to either the next floor up or the next floor down.
- via **portals**: each floor provides (up to) 2 portals. Each portal can take us to a higher floor or a lower floor.
  But which floors? **We don't know!** The portals are pre-defined, but unknown to us in advance.
  The only thing we know is that:
  - if we use the `up` portal from the i-th floor, no subsequent portal will take us to the i-th floor or any floor below it, and
  - if we use the `down` portal, no subsequent portal will take us to the i-th floor or any floor above it. Therefore, a floor might have fewer than 2 portals.

  For example, if floor 1 has a portal to floor 2 (`up` portal), then floor 2 cannot have a `down` portal anywhere.

You need to find the best way to get from floor S to floor T.

The ideal solution avoids using ladders altogether!

Input (`STDIN`) - Output (`STDOUT`)

This problem is interactive, meaning you give instructions to the judge, and it responds with information for you to proceed.

1. The first line contains 3 integers F, S, and T: the number of floors, the starting floor, and the floor where the exit is. You are initially on floor S.
2. As long as you are not on floor T:
   - You print your move: `LADDER UP`, `LADDER DOWN`, `PORTAL UP`, `PORTAL DOWN`
   - The judge prints the floor number x you land on, in the form `OK x` and
   - If you requested a `PORTAL` move on a floor that doesn't have the corresponding portal, the judge will print `SAME`.
3. Once you reach floor T, the game ends and the judge computes your score.

Your score for this game is: $100\% - \frac{\text{number of ladder moves}}{\text{total moves}}$

Example

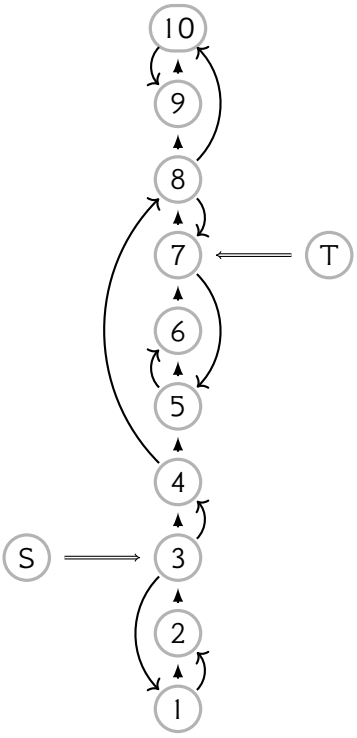
Input (STDIN)

```
10 3 7
```

One possible sequence of moves could be:

Program Output (STDOUT)	Judge Output (STDIN)
PORTAL DOWN	
	OK 1
LADDER UP	
	OK 2
LADDER UP	
	OK 3
PORTAL UP	
	OK 4
PORTAL UP	
	OK 8
LADDER DOWN	
	7

This solution would score $\frac{73}{7} = 57\%$.



Note: To ensure the interactive grader works properly, you must flush the lines you print. This is done as follows:

- In python: `print()` does this automatically

```
...
print("LADDER UP")
...
```

- In C/C++ (`cstdio`): use `printf` as usual

```
...
printf("LADDER UP");
...
```

- In C++ (`iostream`): use `endl`

```
...
cout << "LADDER UP" << endl;
...
```