Important Contest Instructions

Please read the following instructions carefully. They contain important information on how to write your programs, how to run them and how to prepare them for submission.

Language Versions

The judges will be using the following versions for each of our supported languages.

Language	Version
Java	21.0.2
Python	3.12
C	13.2
C++	13.2

Java

Your program source file name must be named **probXX.java** and your class name must be **probXX**, where **prob** is all lower case and **XX** corresponds to the two digit problem number.

- Example class name: `public class prob01`
- Example source file name: `prob01.java`

Java programs can rely on the standard Java library.

Python

We recommend naming your program **probXX.py** (where **XX** corresponds to the two digit problem number).

Python programs can rely on the standard python library.

C / C++

We recommend naming your program source file **probXX.c** or **probXX.cpp** (where **XX** corresponds to the two digit problem number).

C or C++ programs must be submitted as source files and can rely on glibc/libm and the standard headers.

> Download our "C/C++ Developers Guide" from https://hpecodewars.org/downloads

Note

If there is a discrepancy between a data set in the problem statement and the data set file (for either input or output), please consider the data set file to be correct! We will announce any errata as needed.

Program Input

Most programs will require input. You have two options, File Input or Keyboard Input. Some problems may require Directory Input.

File Input

Your program may read input from a file named **input.txt** for all problems. The file will be in the same directory as your program. Use `"input.txt"` as the filename and **do not prefix it with a path**.

> Download our "Guide to data file I/O" from https://hpecodewars.org/downloads

Keyboard Input

Your program may read input from standard in (the keyboard). Do **not** include any prompts in your output. There are two options to provide input to your program via standard in (the keyboard).

- (1) The preferred option is to redirect the contents of a file to standard in of your program.

```
java prob01 < input.txt
python prob01.py < input.txt
prob01.exe < input.txt
```

- (2) Otherwise you may type everything manually, or copy/paste from the data set files.

> Tip: Type `Ctrl-Z <return>` to signal the end of keyboard input.

Directory Input

For some problems, you will need to read from a directory (folder) named `"files"`. The directory will be in the same directory as your program. The student dataset includes a new directory called `"files"`. **MOVE IT** to the same directory as your program.

Use `"files"` as the its name and do not prefix it with a path. The judges will also place the `"files"` directory in their testing directory.

> Download our "Guide to Directory I/O" from https://hpecodewars.org/downloads

Program output

Your program must send the output to the screen (standard out, the default for any print statement). Do **not** include any prompts for input in the output! The only output for your program should be the expected output for the given problem.

Testing your program

Test your program like a judge will with the Python Check Problem script (`checkProb.py`) in the student ZIP file. The script will automatically run your program with each data set for that problem. See `README.txt` in the student ZIP file for more details.

> We strongly encourage you to test your program with all data sets in the `student_datasets` directory of the student ZIP file!

Online Code Submission

Programs will be submitted through the contest site using a text box (with a large character limit, don't worry!) instead of a file upload. You will need to copy and paste the program from your local source code file into the text box. Once you select the problem number and language from the respective drop-downs, you can submit the program for judging.

problem **00**

Sony's Star Wars?

1 points

Input

There is no data file to read in for this (**and only this**) problem.

Output

Print the following 2 lines, exactly as shown:

```
We are pleased to announce the launch of our new game.
It is going to dominate the... hang on I am being handed a note...
```

Discussion

This is the only problem in the packet which does not require you to read data in from a file.

These first two problems are intended to help you check that your setup is working to produce code which will run on the judges' systems. Common problems found every year which will cause the judges to fail your coding for this problem include:

- Including a package reference in a Java class
- Not naming your Java class to match the problem (this is only an issue for Java coders)
- Including libraries not available in the standard library set (this affects everyone -- only vanilla standard libraries are allowed for the contest!)
- Using an IDE (such as BlueJ) which hides the rest of the coding needed for your program to run outside of that IDE
- Hardcoding paths in your code which only exist on your computer (affects everyone)

problem 01

Mockery: You Are Always Right, Master

1 points

Help fix HK-47's vocabulator by running through some name pronunciation drills.

Input

You will receive one name on a single line. It will not contain spaces or punctuation (even if the proper spelling of the name would include them).

```
MASTER
```

Output

Print to the screen the following text. **The only part of the text which should change is the NAME!**

```
Retraction: Did I say that out loud? I apologize, MASTER. While you are a meatbag, I
suppose I should not call you such.
```

Discussion

Be sure to run your solution against **all** of the student data sets. They have DIFFERENT names.

Additional Example

Input 2	Output 2
BASTILA	Retraction: Did I say that out loud? I apologize, BASTILA. While you are a meatbag, I suppose I should not call you such.

problem 02

Flash Forward

2 points

The Flash has run forward in time too often and has started diplicating himself. This is only a problem if he duplicates himself an odd number of times. If he has an even number of duplicates he can keep skipping forward in time, else he will have to abandon his mission to fix the Space-Flash-Time-Continuum (yes, that is a real thing).

Input

You will receive a list of integers (no integer will be larger than what an int-32 can hold, integers may be positive or negative), one per line. The minimum count of integers you might receive is four (4). When you see zero (0) on a line by itself, input has terminated. (Do not include the zero in the data set).

```
2
-2
4
8
19
-100
15
0
```

Output

Figure out if all of the integers are even. If any odd numbers are found print the odd numbers found (in the order found in the input) and the statement FLASH FAILED. If the list contains only even numbers, output: FLASH FORWARD.

```
19 is odd
15 is odd
FLASH FAILED
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input 1	Output 1
2 4 6 22 -18 90 102 0	FLASH FORWARD

problem 03

Jarvis has ... issues

2 points

Jarvis is malfunctioning and asking Tony for repairs on its math co-processor. Help Tony write a quick program to test the repairs on the math operations the A.I. is producing so he can get back into the battle.

Input

You will receive up to four lines of data in the format of: `A B OP OUT`
END on a line by itself ends input.

Examples:

```
100 7 DIVIDE 5.0
7 7 MULTIPLY 49.0
3 7 SUBTRACT 6.0
3 7 ADD 12.0
END
```

The value for A or B can be negative.

Output

Convert all input numbers to decimal on output. All output numbers need to be **truncated** to 1 decimal place.

If a miracle happened and Jarvis was correct, output is in the format of:

```
OUT is correct for A OP B
```

If Jarvis is incorrect, output is in the format of:

```
A OP B = correct answer, not OUT
```

Examples:

```
100.0 / 7.0 = 14.3, not 5.0
49.0 is correct for 7.0 * 7.0
3.0 - 7.0 = -4.0, not 6.0
3.0 + 7.0 = 10.0, not 12.0
```

Jarvis has ... issues

continued

Discussion

Truncate means to *cut off*. E.G. if your output is 3.9572, if you need to truncate to two decimal places your output should be 3.95. If you need to truncate to one decimal place, your output should be 3.9. Truncate **does not mean** to round.

Your program will need to be able to handle OP codes for: MULTIPLY, DIVIDE, ADD and SUBTRACT

Operations should be carried out left to right A to B as follows:

```
A * B (A multiplied by B)
A / B (A divided by B; note Jarvis is still functional enough to not give divide by
zero problems)
A + B (A added to B)
A – B (B subtracted from A)
```

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input 1	Output 1

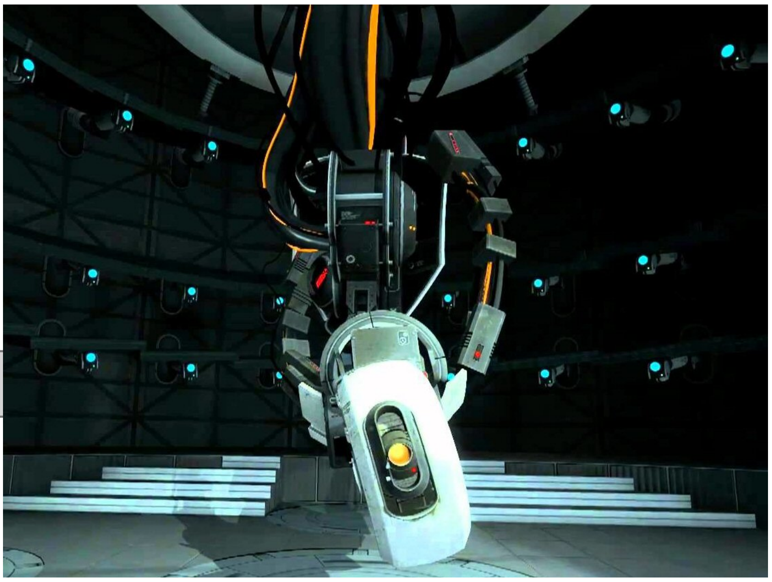
`-33 9 SUBTRACT 22.0`
`1 0 MULTIPLY 0.0`
`1000 2000 ADD 4000.0`
`-55 -5 DIVIDE 11.0`
`END`
	`-33.0 - 9.0 = -42.0, not 22.0`
	`0.0 is correct for 1.0 * 0.0`
	`1000.0 + 2000.0 = 3000.0, not 4000.0`
	`11.0 is correct for -55.0 / -5.0`

problem 04

For Science

3 points

It's been a long time. And we have science to do. I have plans, and I suppose I need you. Today we are testing numbers that could be prime. Find those numbers, don't waste my time. For Science. You Monster.



Input

You will receive a single positive, non-zero, int-32 on one line.

```
4
```

Output

Determine if the number COULD be prime by adding or subtracting one to it. If the number is already prime, say so. If the number is not prime, and cannot be made prime by adding or subtracting one, say it is a bad specimen.

```
4 could be prime!
```

Discussion

A prime number is defined as any positive number which can only be evenly divided by 1 and itself, excluding 1 and zero.

In the first example, +1 or -1 to 4 will yield 3 or 5, both of which are prime. So it could become prime given the test parameters. In the further examples below -1 to 14 will yield 13, which is prime (+1 would yield 15, which is not prime). In the case of 21, it is not prime, +1 yields 22, and -1 yields 20, neither of which are prime, so it is a bad test specimen.

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
7	7 is already prime
14	14 could be prime!
21	21 is a bad test specimen
1	1 could be prime!

problem 05

Mirrored Darkness

3 points

Help guide Rey away from the Darkness. With ASCII. (It's complicated.)

Input

You will receive two lines of input. The first line will contain a single character, which will be the reflection point. The second line will be the line to move around the reflection point.

```
7
abcdefghijklmnopqrstuvwxyz9876543210
```

Output

The reflection should cut and swap positions of the string parts at the point of reflection. CaSe MatTeRs! If the reflection point is given as `Q` that **means** `Q`, not `q` or `qq` etc.

```
65432107abcdefghijklmnopqrstuvwxyz98
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
W xXxXxXxXxXxXWwwwwwwwj	wwwwwwwjWxXxXxXxXxXxX

problem 06

Doh!

4 points

Homer is visiting from another dimension to give you some calculations Lisa made to finish the portal into their dimension (it's a long story).

Unfortunately, Lisa's back-of-the-envelope estimates aren't lining up as expected.

NASA thinks it may be because Lisa was doing quick counts on her fingers, and she only has 4 on each hand, that would mean the numbers she is giving are in **quaternary** instead of **decimal!**

Help NASA automate a program to convert quaternary (base-4) numbers into decimal (base-10) numbers so that work can continue on the portal.

Input

You will receive a list of **quaternary** numbers, one number per line. Input ends at triple zeros. You will not be given zero (as a quaternary number) to be translated into decimal. The largest quaternary number you will be expected to handle will be 122230333000. Example:

```
1111
21
12132
10231122013
122230333000
000
```

Output

Output the value of the quaternary numbers, one per line (ignoring the 0 line).

```
85
9
414
1234567
7000000
```

Discussion

Reminder: have you run your solution against **all** of the student data sets? Hint: some of the values you may get as input may be larger than what an int-32 variable can hold.

Additional Examples

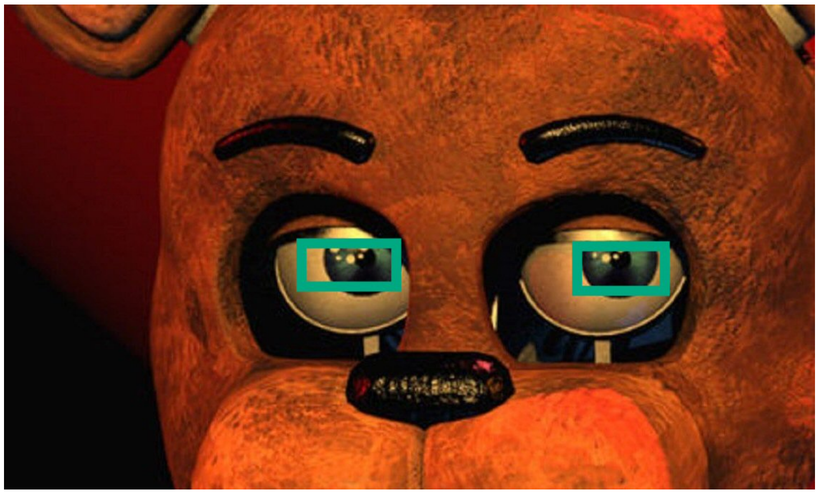
Input	Output
22 23 30 31 32 33 100 000	10 11 12 13 14 15 16

problem 07

Security Office

5 points

The New Guy™ in the security office watching Freddy Fazbear's Pizzeria wandered out to check the floor before you could update him on the *situation*. Use your computer-generated overhead grid to warn him if an animatronic is in the room with him!



Input

You will receive a grid made up of lines of ASCII characters. The grid will be between 5-40 lines tall, and 5-80 characters wide. Chairs will be marked on the grid with a (C), Tables with a (T), arcade machines with an (M), and walls with (W) characters. If an animatronic is found by the security system, it will be marked with an (A). All other empty spaces on the grid will be marked with a period (.)

```
...MM........C.W....
...MM........C.W....
...............W..T.
...MM........C.W....
.A.MM........C.W....
...............W....
WWWWWWWWWWWWWWWW....
```

Output

If an animatronic shows on the grid with an (A) character, quickly tell New Guy™ where it is and tell him to run! If an animatronic isn't spotted say that the coast is clear and he is lucky. The coordinates are in (column, row) format.

```
Get out of there! Danger at (1, 4)
```

Additional Examples

The upper left corner of the grid is (X=0,Y=0). The grid moves only in the positive direction down and to the right from that corner.

Reminder: have you run your solution against **all** of the student data sets?

Input	Output
WWWWW ...TC .M... .M... WWWCT	You are lucky. Get back to the office NOW!

Line Must Go Up

Stock holders do not like it when the line goes down. That means they lose money. You, being a gullible Gen-"A.I.", have been asked to flip lines going negative to show positive growth. Turns out ethical controls on a bot are easy to trick. Who knew?

Input

The first line of your input will be an integer showing the count of lines you will encounter in the rest of the input, which will be the # symbol in a wave pattern. There will be no flat parts to the waves. Max Size of a wave pattern is 80x80.

```
7
   #
  # #         #
 #   #       # #
#     #     #   #
       #   #     #
        # #
         #
```

Output

The output should be the inverted (flipped) wave.

```
         #
        # #
       #   #     #
#     #     #   #
 #   #       # #
  # #         #
   #
```

Discussion

You are a pretty basic Gen-"A.I."
Your training data only includes instructions about flipping the lines over the x-axis by 180 degrees. That's all you know how to do.

Additional Examples

Input

```
8
#                 #
 #   #           # #
  # # #         #   #     #
   #   #       #     #   # #
        #     #       # #   #
         #   #         #
          # #
           #
```

Output

```
           #
          # #
         #   #         #
        #     #       # #   #
   #   #       #     #   # #
  # # #         #   #     #
 #   #           # #
#                 #
```

Cerberus Funds

Miranda is looking for some missing System Alliance funds a Cerberus hacker managed to steal but couldn't get away with. If she can find the funds she can help Shepard appropriate the funds to help with the upkeep on the Normandy. You will need to scan the data in your exfiltrated Alliance files directory to search for the *"liberated"* money using the signal marker the hacker left behind. If you find any of the missing funds, total them up and record their transaction IDs to give to Mordin to yoink later.

Input

Your Cerberus sources have sent you a lead on the *liberated* funds. The lead will be in the format of:

```
Signal
Date-Start
Date-Stop
```

Example:

```
Gamma5Jay
2187-01-17
2189-02-03
```

The dates will align with files named 2189-01-17.txt, 2189-02-03.txt and additional files with dates that fall inside the given range. (Just because a date is given in the range does not mean a file with that date exists).

The contents of the files will always be in the format of:

```
transaction-ID,date-time,action-code,data-in,data-out,operator-code
```

You will be looking for lines with operator codes changed to match the signal our hacker left behind. The funds available to *liberate* will be recorded in the data-out field.

Example data line with a signal matching the input:

```
712D036,21890117130959,FUNDADJ,-41239.02,41239.02,Gamma5Jay
```

Cerberus Funds

continued

Output

Output should list all of the transaction IDs found, ordered in ascending order, and then on the last line the total amount of funds available to *liberate*. As the funds are all listed in the records using 2 decimals points, that is also how your program should output the total.

Example:

```
05FXT3N
2IU3AJ15AF74
49KL39V4
4XI3X7624YIR
5PFVOT1O21CK
66OILUP1X5C
Funds Found: 18990080.04
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
Section31 2187-04-12 2188-12-03	DQ1GX1 Funds Found: 9716204.38