UAB CIS High School Programming Contest

April 30, 2016

THESE ARE THE OFFICIAL CONTEST QUESTIONS!

Each problem in this packet contains a brief description, followed by three example test cases of a successful implementation.

The example input and output shown for each question should be regarded only as examples. We will test your programs on the examples provided with each problem, as well as several other test cases generated by the judges. Be sure to follow the input and output formatting exactly, printing each output answer on its own line. Extraneous or malformed output will result in a failed submission.

If your program fails, a result will be returned by the submission system stating the counterexample test case that caused your program to be judged incorrect. Each incorrect answer will be incur a 20 minute time penalty, but this penalty will only be applied to your score if the problem is eventually answered correctly. As noted in the rules, the overall ranks will be determined first by the number of problems completed, and then by your time score (including penalties).

Please pay close attention to the directions in each problem description. In some cases, assumptions are stated about the limitations of the input, which are designed so that you do not have to consider difficult cases or perform input validation.

If you get stuck on a problem, you are encouraged to jump around and try different problems, as an eventual correct answer on a problem is better than no answer at all.

1 Monochrome

A digital picture is comprised of a series of pixels. In a color picture, each pixel is represented by three components of red, green, and blue (RGB). Each component has a value n, $0 \leq n \leq 255$. In grayscale images, a pixel is represented by its luminance l, $0 \leq l \leq 255$. To convert from color to grayscale, we can multiple each color component by its luminance constants and sum the results. Luminance constants for R, G, and B are 0.21, 0.71, and 0.08. For example, the RGB color $125, 10, 120$ would be 43 (round$(125 * .21 + 10 * .71 + 120 * .08)$). Write a program that converts RGB colors to grayscale.

The input consists of a line that contains the RGB values separated by a space. The output is the grayscale value. The program terminates after the computed grayscale value is 0.

Sample Input/Output

Input	Output
125 10 120	43
255 255 255	255
17 22 125	29
0 0 0	0

2 Pairwise sums

Given a list of unique positive integers, calculate the total number of all possible pairs whose sum yield a given target. For example, in a list of $[4, 6, 77, 5, 88, 219, 828, 79]$ and a target number 83, the program will find the pairs $[77, 6]$ and $[4, 79]$.

The input consists of two lines. The first line specifies the target number, the second line contains a list of unique positive integers separated by space. The program prints the total number of pairs equaling the target sum. The program terminates when the target number is 0.

Sample Input/Output

Input	Output
83 4 6 77 5 88 219 828 79	2
10 1 2 3 4 5 6 7 8 9 10	4
0	

3 Smallest possible number

Concatenate a sequence of numbers, so that the concatenated number is as small as possible. For example, the sequence 12 872 9 122 would yield the solution 121228729.

The input consists of a line containing positive integral numbers separated by a space. The output prints the smallest possible number that can be produced by concatenating the entered numbers. The program terminates, as soon as the smallest number is 0.

Sample Input/Output

Input	Output
12 872 9 122	121228729
1 12 112 111	111111212
0	0

4 Only Hundred

Given the numbers from 1 to 9 (in ascending order). Devise a program that adds either a + or a − or nothing (meaning concatenation with previous number) in front of every number. If computed, the result should be 100. For example, $1 + 2 + 34 - 5 + 67 - 8 + 9$ yields 100.

The program takes no input, and prints out all possible solutions. **Output should be printed in ascending string sorted order!**

Sample Input/Output

Input	Output
	$1 + 2 + 34 - 5 + 67 - 8 + 9$ ...

5 Sum of Fractions

Any fraction $0 < \frac{x}{y} < 1$ can be rewritten as sum of fractions where all numerators are 1 and all denominators are distinct. For example $\frac{3}{13}$ could be expressed as $\frac{1}{5} + \frac{1}{33} + \frac{1}{2145}$.

Write a program that reads in the numerator *num* and denominator *denom* of a fraction and that prints a sequence of fractions whose sum yields $\frac{num}{denom}$. The numerators of the fractions in the sequence must be 1 and all denominators must be distinct.

The input consists of two integers, numerator and denominator, separated by a space. The output consists of the fractions separated by space. The program terminates after the input is $\frac{0}{0}$.

Sample Input/Output

Input	Output
3 13	1/5 1/33 1/2145
1 2	1/2
3 12	1/4
0 0	

6 Cipher

A simple cipher could be implemented by reformatting a text to fit within a given amount of columns. Then we could encrypt the message by systematically swapping two columns or two rows. For example the text:

UAB students are excellent programmers

could be reformatted into column width of 8, giving:

```
UAB stud
ents are
 excelle
nt progr
ammers
```

Then we swap the first and seventh column, denoted by the triple (c 1 7), followed by swapping the second and fourth row, denoted by the triple (r 2 4). This yields the following arrangement:

```
uAB stUd
gt pronr
lexcel e
rnts aee
 mmersa
```

The result is an encrypted message:

uAB stUdgt pronrlexcel ernts aee mmersa

The input consists of a line containing the text, followed by a line containing the column width, followed by a line indicating the number of swaps, followed by n lines of swap codes. A swap code is a triple abc where a indicates whether it is a row or column swap, and b and c are the column/row numbers to swap (starting at 1).

Sample Input/Output

Input	Output
UAB students are excellent programmers 8 2 c 1 7 r 2 4	uAB stUdgt pronrlexcel ernts aee mmersa