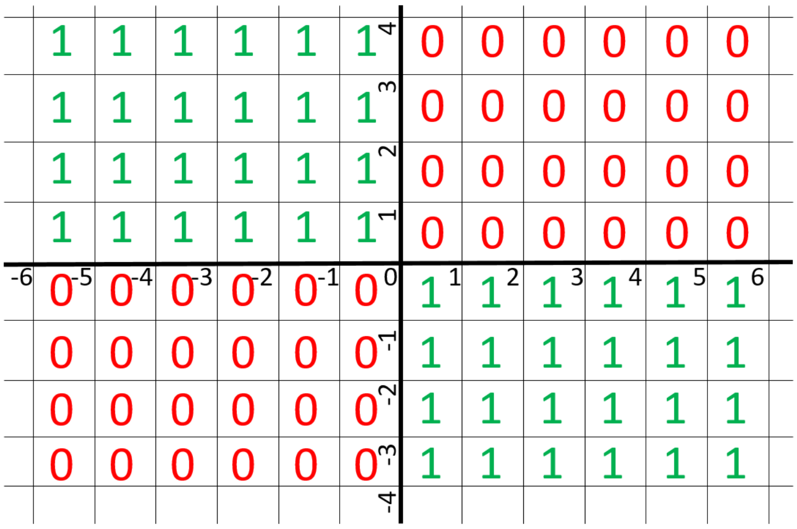
LAB-PRAC-06_LOOPS

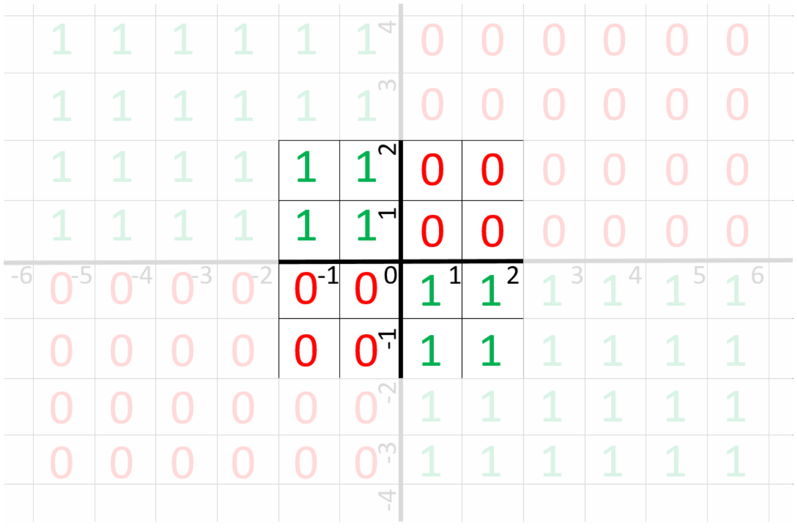
Fill in the Square (p1v1d1)

Fill in the Square [10 marks]

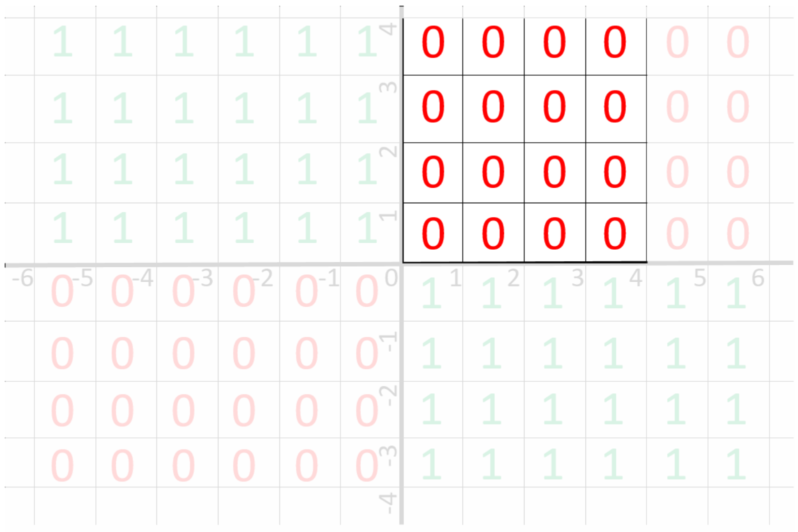
Problem Statement
The entire Cartesian plane has been divided into unit squares. Squares in the 1st and 3rd quadrant are filled with 0 and squares in the 2nd and 4th quadrant are filled with 1 as shown below.



You are given the side length (**always a non-negative integer**) of a square and the **integer** coordinates of its bottom left corner. You have to show us in your output, what that square looks like. For example if the side length is 4 and the bottom-left corner is (-2,-2), then the square looks like



In this case you have to print
1100
1100
0011
0011
Notice that there are not spaces anywhere and no extra new lines. As another example, if the side length is 4 and the bottom-left corner is (0,0), then the square looks like



In this case you have to print
0000
0000
0000
0000

Caution

1. Remember, side length can be any non-negative integer, not just 4 as in the visible test cases. Coordinates can be negative or positive integers and can even be zero.
2. Be careful about extra/missing lines and extra/missing spaces.
3. Be very careful, even though the evaluation may give you marks for extra spaces and newlines, the autograder will give you zero marks for any extra spaces or new lines.

EXAMPLE:
INPUT
4 -2 -2

OUTPUT:
1100
1100
0011
0011

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
4 0 0	0000

	0000 0000 0000
4 -2 -2	1100 1100 0011 0011
6 -2 0	110000 110000 110000 110000 110000 110000
3 0 -2	000 111 111
1 5 -5	1
2 -1 -1	10 01

Pretty Numbers (p1v2d1)

Pretty Numbers [20 marks]

Problem Statement
Given a **strictly positive integer** N, on the first line, output K defined as the sum of the digits of N. Then output the smallest number strictly greater than N, that is a perfect square, and also divisible by K. For example, if N = 13, then the sum of digits of N is 1 + 3 = 4. The smallest perfect square divisible by 4 and strictly greater than 13 is 16, so the output should be

```
4
16
```

Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. Although it is not absolutely needed, you may use the log10() function by including math.h that can help you calculate the length, i.e. number of digits, of an integer.
3. Please note the unusual grading scheme for this question. The grading scheme is described below.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 25% weightage whereas the second line carries 75% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
3	3 9
15	6 36
81	9 144
6789	30 8100
1000	1 1024
10000	1 10201

Block Cipher (p1v3d1)

Block Cipher [20 marks]

--

Problem Statement
You will be given a **long integer** N and another **integer** k as input. You have to take the number N and divide it into blocks of length k. For example if the number is 1999567890 and k = 3, then the blocks would be
(1)(999)(567)(890). Note that the last block has only one digit in it.
Given this block structure, we derive an encrypted number as follows: take each block and add the last digit of that block to that block. Take the last k digits of the new number we get as the sum to get the encrypted block. For example, in the above case, we have k = 3, so
890 + 0 = (890)
567 + 7 = (574)
999 + 9 = 1(008)
1 + 1 = (2)
The encrypted number is formed by concatenating all the encrypted blocks i.e.
2008574890

On the first line, print the number of blocks in the integer. In the above example, there are 4 blocks in the number. On the second line print the encrypted number.
Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. Please note the unusual grading scheme for this question. The grading scheme is described below.

HINTS: Please include the math.h library in your code in case you feel it helps you solve this problem. The log10() function available through the math.h library may help you calculate the number of digits in a number.

EXAMPLE:
INPUT
1999567890 3
OUTPUT:
4
2008574890

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 25% weightage and the second line carries 75% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
123456 1	6 246802
112233445566 2	6 122436486072
999888777666 3	4 8896784672
1234567890123456789 3	7 2238574890126462798
123456789 9	1 123456798
999999999999 6	2 8000008

The Fibonacci Facade (p2v1d1)

The Fibonacci Facade [10 marks]

Problem Statement
Recall that the Fibonacci numbers are defined as follows
F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2), for n > 1

Given an **integer n**, print the Fibonacci triangle described below in the example. The triangle should have n rows

1. The first row should print F(0)
2. The second row should print F(0) F(1) (i.e. the first two Fibonacci numbers **separated by a single space**)
3. The third row should print F(0) F(1) F(2) (i.e. the first three Fibonacci numbers each separated by a single space)
4. ... and so on

There should be no space after the last number in any line. There should be no extra newline after the last line of the output. If n < 1, then your program should print "INVALID INPUT" (without quotes).

Caution

1. Fibonacci numbers can get very large. Even though n will fit inside int variables, **use long variables** to do all your computation and all your printing.
2. Be careful about extra/missing lines and extra/missing spaces.
3. Note that there is **no trailing space** at the end of the last number F(j-1). If your output has trailing spaces, you will receive no marks as there is no partial grading in this question.
4. Note that there is **no trailing new line** after the last line of the output.
5. Be very careful, even though the evaluation may give you marks for extra spaces and newlines, **the autograder will give you zero marks** for any extra spaces or new lines.
6. Take care of capitalization and spelling mistakes.

EXAMPLE:
INPUT
5

OUTPUT:
```
0
0 1
0 1 1
0 1 1 2
0 1 1 2 3
```

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases

All Test Cases (Visible + Hidden)

Input	Output

5	0
	0 1
	0 1 1
	0 1 1 2
	0 1 1 2 3

-12	INVALID INPUT
0	INVALID INPUT

10	0
	0 1
	0 1 1
	0 1 1 2
	0 1 1 2 3
	0 1 1 2 3 5
	0 1 1 2 3 5 8
	0 1 1 2 3 5 8 13
	0 1 1 2 3 5 8 13 21
	0 1 1 2 3 5 8 13 21 34

50	0
	0 1
	0 1 1
	0 1 1 2
	0 1 1 2 3
	0 1 1 2 3 5
	0 1 1 2 3 5 8
	0 1 1 2 3 5 8 13
	0 1 1 2 3 5 8 13 21
	0 1 1 2 3 5 8 13 21 34
	0 1 1 2 3 5 8 13 21 34 55
	0 1 1 2 3 5 8 13 21 34 55 89
	0 1 1 2 3 5 8 13 21 34 55 89 144
	0 1 1 2 3 5 8 13 21 34 55 89 144 233
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887

	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049

30	0
	0 1
	0 1 1
	0 1 1 2
	0 1 1 2 3
	0 1 1 2 3 5
	0 1 1 2 3 5 8

	0 1 1 2 3 5 8 13
	0 1 1 2 3 5 8 13 21
	0 1 1 2 3 5 8 13 21 34
	0 1 1 2 3 5 8 13 21 34 55
	0 1 1 2 3 5 8 13 21 34 55 89
	0 1 1 2 3 5 8 13 21 34 55 89 144
	0 1 1 2 3 5 8 13 21 34 55 89 144 233
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811
	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229

Stream AM GM (p2v2d1)

Stream AM GM [20 marks]

Problem Statement
You will be given a stream of integers, **both positive as well as negative**. Your job is to calculate the arithmetic mean and the geometric mean of the numbers in the stream till -1 is encountered (excluding -1). Print the AM and GM **rounded off to 4 decimal places** on **two different lines**.

If the GM of the numbers turns out to be complex number (since the product of the numbers, not including the last -1 is negative) then print the AM as is but in place of GM, print "INVALID INPUT" (without the quotes). If the stream is empty, i.e. the first number is itself -1, then print the following words "EMPTY STREAM" (without the quotes) on both lines.

Caution

1. Be careful about capitalization and spelling errors.
2. Be careful about extra/missing lines and extra/missing spaces.
3. Note that the stream may contain positive as well as negative numbers. However, you only have to calculate AM and GM of the numbers till -1 is encountered (not including the -1).
4. Although the stream input will fit inside int variable, use **double variables and double typecasts** to do all your calculations.
5. Careful: the GM is always a positive number

6. Dont forget to include math.h in your code if you need math functions

----
EXAMPLE:
INPUT
1 2 3 4 5 6 7 8 9 10 -1

OUTPUT:
5.5000
4.5287

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 -4 -1	-1.0000 INVALID INPUT
1 2 3 4 5 6 7 8 9 10 -1	5.5000 4.5287
-1	EMPTY STREAM EMPTY STREAM
9 9 9 9 9 9 9 9 9 -1 100 100 100 100	9.0000 9.0000
1 2 3 4 5 6 7 8 9 0 9 8 7 6 5 4 3 2 1 -1 1000 1000 10000 100000	4.7368 0.0000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 -1	50.5000 37.9927

Int on Int (p2v3d1)

Int on Int [20 marks]

Problem Statement

You will be given two **non-negative integers** N, k, with the guarantee that both are not zero simultaneously. Your job is very simple. In the first line, print the sum 1^k + 2^k + 3^k + 4^k + 5^k. In the second line, print the sum 1^k + 2^k + ... + N^k. We want your output to be printed as an integer with all its digits intact - i.e. no approximations.

Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. Even though N, k will fit inside int variables, use **long variables and long typecasts** in your calculations.
3. Remember, datatypes like float and double can distort the numbers they store inside them.
4. For N = 0, the summation 1^k + 2^k + ... + N^k is empty and is value should be printed as 0.

INPUT:
N k
OUTPUT:
1^k + 2^k + 3^k + 4^k + 5^k
1^k + 2^k + ... + N^k

EXAMPLE:
INPUT
1 1

OUTPUT:
15
1

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
1 2	55 1
5 2	55 55
30000 0	5 30000
0 10	10874275 0
14 11	53201625 7012409924625

12 10	10874275 102769130750

Bejewelled Brooch (p3v1d1)

Bejewelled Brooch [10 marks]

Problem Statement
You have to print the following beautiful pattern, given two **strictly positive integers** M and N. The first input will be M and then N will be provided after a space

```
1 3
0000000
0#0#0#0
0000000

2 2
00000
0#0#0
00000
0#0#0
00000
```

Thus, as you see, there will be M special rows where 0 (zero) and # alternate. There are N # signs present in each of these special rows and each # sign has one 0 sign to the left and to the right. Also, each special row has a row filled with 0 signs below and above it.

Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. There should be no trailing spaces at the end of each line or trailing new lines.
3. Be very careful, even though the evaluation may give you marks for extra spaces and newlines, **the autograder will give you zero marks** for any extra spaces or new lines.

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
1 1	000 0#0

	000
1 3	0000000
	0#0#0#0
	0000000
2 3	0000000
	0#0#0#0
	0000000
	0#0#0#0
	0000000
1 10	000000000000000000000
	0#0#0#0#0#0#0#0#0#0#0
	000000000000000000000
10 1	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
	0#0
	000
4 6	0000000000000
	0#0#0#0#0#0#0
	0000000000000
	0#0#0#0#0#0#0
	0000000000000
	0#0#0#0#0#0#0
	0000000000000
	0#0#0#0#0#0#0
	0000000000000

Mobile Mixup (p3v2d1)

Mobile Mixup [20 marks]

---

Problem Statement
On planet Vormir, where last week's aliens resided, mobile numbers can be upto 18 digits long and are not of fixed length i.e. some mobile numbers may have just 5 digits and others may have 18 digits. Facebook recently opened its operations in Vormir but its messaging system broke down and transmitted mobile

numbers in reverse (Mark Zuckerberg had to apologize in front of the Vormir parliament too).

Vormir aliens know that a real mobile number may end with zero but never start with a zero, just like Earth mobile numbers. However, when the mobile numbers were reversed, the zeros at the end were lost. You will be given two numbers a **long integer** M containing the reversed mobile number and an **integer** n tell you how many digits that mobile number has.

In the first line of your output, give the first 3 digits of the original (unreversed) mobile number (no spaces between the digits) and in the next line, give the complete mobile number.

Caution

1. Use long variables to input and process the reversed and the original mobile numbers. They may not fit inside an int variable.
2. Be careful about extra/missing lines and extra/missing spaces.
3. The partial grading scheme in this question is unusual. See below for grading policy for this question.

HINTS:

1. The function log10(n) available from math.h gives you back the base 10 logarithm of a number n. This function can help you in finding the number of digits in an integer n. Include math.h in your code to use this.

INPUT:
M n

OUTPUT:
First three digits of original mobile number (no spaces)
Complete original mobile number

EXAMPLE:
INPUT
9 8

OUTPUT:
900
90000000

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. The first line carries 20% weightage and the second line carries 80% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
1234567891 10	198 1987654321
9 8	900 90000000
1122334455 18	554 554433221100000000
123456789123456789 18	987 987654321987654321
102030405 16	504 5040302010000000
9801762354 10	453 4532671089

Primes are in C (p3v3d1)

Primes are in C [20 marks]

Problem Statement
Prime factorization is an important topic in analytic and computational number theory. Any positive integer can be expressed as product of prime numbers. You will be given a **strictly positive integer** N, and you have to print two quantities on **two separate lines**

1. the number of divisors of N (including 1 and the number itself)
2. the sum of all prime divisors of N (including the number itself if N is itself prime)

Caution

1. In the first line, we want the number of all divisors (including 1 and the number itself). We do not care if the divisors are prime or not in this line.
2. In the second line, we want the sum of only the prime divisors of N (including the number itself).
3. 1 is not considered a prime number but 2 is considered a prime number
4. Be careful about extra/missing lines and extra/missing spaces.

EXAMPLE:
INPUT
4

OUTPUT:
3
2
Explanation 4 has 3 divisors 1, 2, 4, out of which only 2 is prime so sum of primes is just 2.

Grading Scheme:

Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
9	3 3
4	3 2
1	1 0
37	2 37
6578932	12 96768
47379084	36 10961

Towering Numbers (p4v1d1)

Towering Numbers [20 marks]

Problem Statement
Given a **strictly positive integer** N >= 1, your job is to print a number pyramid as show below. Suppose N = 5, then the pyramid should look like

```
5
454
34543
2345432
123454321
```

The first line has four spaces and then a 5, then no extra spaces. The second line has three spaces and then 454 (no spaces between the numbers) and then no extra spaces, and so on. There is no trailing new line after the last line is over.

Suppose N = 3, then the pyramid should look like

```
3
232
12321
```

The first line has two spaces and then a 3, then no extra spaces. The second line has one space and then 232 (no spaces between the numbers) and then no extra spaces, and so on. There is no trailing new line after the last line is over.

Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. Be very careful, even though the evaluation may give you marks for extra spaces and newlines, the autograder will give you zero marks for any extra spaces or new lines.

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
5	5 454 34543 2345432 123454321
3	3 232 12321
1	1
6	6 565 45654 3456543 234565432 12345654321
9	9 898 78987 6789876 567898765 45678987654 3456789876543 234567898765432 12345678987654321
4	4 343 23432 1234321

A Run of One (p4v2d1)

A Run of One [20 marks]

--

Problem Statement
We will give you a sequence of integers which are guaranteed to be either 0 or 1. The sequence will be ended by a -1. You have to output three quantities as your output on **three different lines**

1. Total number of integers in the sequence, not including the final -1
2. Total number of zeros in the sequence
3. The maximum length of a continuous sub-sequence of 1s in the sequence

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.
2. The sequence may be empty (first number may itself be -1) or else the sequence may contain only one integer.
3. Please note the unusual partial grading scheme for this question, given at the bottom.

HINTS: Recall we learnt about running counters and running sums in our earlier discussions. Those may help you solve this problem.

--

EXAMPLE:
INPUT
0 1 0 0 1 1 1 0 1 1 -1

OUTPUT:
10
4
3
Explanation: There are 10 integers in the sequence (not including the final -1), there are 4 zeros in total, and the continuous sub-sequences containing only 1s are the following (marked using brackets)
0 (1) 0 0 (1 1 1) 0 (1 1) -1
Of these, the longest such sequence is of length 3.

--

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are three lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first two lines carry 12.5% weightage each and the last line carries 75% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
0 1 0 0 1 1 1 0 1 1 -1	10 4 3
1 0 1 0 -1	4 2 1
-1	0 0 0
1 -1	1 0 1
1 1 0 0 1 1 1 0 0 0 1 1 1 1 -1	14 5 4
0 0 0 0 0 0 0 0 0 0 -1	10 10 0

Where are the primes- (p4v3d1)

Where are the primes? [20 marks]

Problem Statement
An interesting result in number theory states that the difference between any strictly positive integer and the number formed by reversing its digits is always divisible by 9, hence the difference can never be prime. However, Mr. C is more interested in prime numbers, and hence decides to find a prime number just greater than this difference.

Given a **strictly positive integer** N as input, find the least prime number - P that is greater than the absolute difference between N and the number formed by reversing its digits. Print the absolute difference and P in two separate lines as output.

Caution

1. Be careful about extra/missing lines and extra/missing spaces.
2. 1 is not considered a prime
3. The number formed by reversing the digits of a number may be greater or smaller than the number itself.
4. You may use the abs function to calculate the absolute value of an integer. Include stdlib.h in case you do wish to use this function.
5. We assure you that the numbers we give you and the outputs, will always fit inside int.

INPUT:

OUTPUT:

EXAMPLE:
INPUT
37

OUTPUT:
36
37

Explanation: The number formed by reversing 37 is 73. The difference between 37 and 73 is 36. The prime number just larger than 36 is 37 itself.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
37	36 37
4	0 2
10001	0 2
12314000	12272679 12272681
176000154	275000517 275000543
543786892	245099547 245099551