The 6th Annual University of Akron Programming Competition

presented by

The University of Akron Computer Science Department

Association for Computing Machinery Student Chapter

April 9, 2015

Rules:

1. There are six questions to be completed in four hours.
2. C, C++, and Java are the only languages available.
3. Data is read from Standard Input and output is sent to Standard Output. Do not prompt for input values in the code you submit to be judged. Do not attempt to read from or write to any files.
4. All programs are submitted as source code only. Submitting compiled information (binary, .class) will result in a Compilation Error penalty.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation).
6. The input to all problems will consist of multiple test cases. The Sample Input listed on the problem description is not intended to be a comprehensive input set. The input is guaranteed to adhere to the descriptions in each problem; you do not need to check for invalid input.
7. The output to all problems should conform to the format shown in the Sample Output for that problem, including but not limited to capitalization, whitespace, etc.
8. Use of personal electronics during the competition is cause for disqualification. Cell phones must be turned all the way off (not silent mode, not airplane mode, etc.). You may use any written notes, books, or reference materials you bring with you.
9. Programming style is not considered in this contest. You can code in any style or with any level of documentation your team prefers.
10. All communication with the judges will be handled through PC2. All judges decisions are final.

A: Operations

Thomas is really, really bad at math. He is so incredibly bad at math that he can't compute simple operations. However, he is a pretty handsome guy and his charisma roll was high, so he's managed to convince you to help him out.

In particular, Thomas needs you to tell him the answer to a computation of the form x *op* y, where x and y are single digit numbers, and *op* is one of either $+$, $-$, $*$, or $/$. For each given operation Thomas only cares about the truncated integral result. Thus, the result of $5/2$ is 2, not 2.5. Help Thomas out!

Input

The input consists of three characters x, p, and y. Both x and y are integers ($1 \leq x, y \leq 9$). And p is either $+$, $-$, $*$, or $/$.

Output

Output the result of the integer operation.

Sample Input	Sample Output
`1 + 2`	`3`

Sample Input	Sample Output
`9 * 9`	`81`

Sample Input	Sample Output
`1 / 9`	`0`

Sample Input	Sample Output
`2 - 3`	`-1`

B: Fibonacci

Eric really likes his recurrence relations. One of his favorites is the Fibonacci recurrence relation. That is, $F_n = F_{n-1} + F_{n-2}, F_0 = 0, F_1 = 1$. Of course, the initial conditions for Fibonacci can take one more than just those values. In fact, the Fibonacci initial conditions are regularly stated as either $(0, 1)$, $(1, 1)$ or $(1, 2)$. You can imagine that initial conditions that maintain the Fibonacci sequence are going to faithfully generate the sequence.

But this got Eric thinking. What if we wrote the three separate recurrence relations down with the three more common initial conditions:

$a_n = a_{n-1} + a_{n-2}, a_0 = 0, a_1 = 1,$

$b_n = b_{n-1} + b_{n-2}, b_0 = 1, b_1 = 1,$

$c_n = c_{n-1} + c_{n-2}, c_0 = 1, c_1 = 2.$

Then, mix up the recurrences to make them all interdependent like this:

$a_n = b_{n-1} + c_{n-2}, a_0 = 0, a_1 = 1,$

$b_n = c_{n-1} + a_{n-2}, b_0 = 1, b_1 = 1,$

$c_n = a_{n-1} + b_{n-2}, c_0 = 1, c_1 = 2.$

While maintaining the same initial conditions. This got Eric really wondering, what would be the nth triple $(a_n,\ b_n,\ c_n)$ of this interdependent relation? Well, Eric happens to be a busy guy, so he doesn't have time to figure it out on his own, so he's hired you to do the job.

Input

The input consists of one integer n $(1 \leq n \leq 40)$.

Output

Output the resulting triple, $a_n\ b_n\ c_n$.

Sample Input	**Sample Output**
`6`	`8 10 8`

C: The Wheel Apparatus

Kacey has a peculiar apparatus of wheels that she wants to share with you. The apparatus rests on the ground with wheels of different radii. Each wheel has an associated crank to turn the associated wheel. With one revolution of a crank for a given wheel the apparatus moves a distance equal to the radius of the wheel. Kacey has spent a lot of time working on this apparatus of hers, and she's very proud of it. She's managed to grease the cranks in a fashion that the effort taken to crank one revolution is indistinguishable between all the wheels.

She's a great mechanic, and not bad at math either, but she can't tire herself out moving this thing. She figures it'd be good to know the minimum number of cranks it would take to move the apparatus exactly a distance t. Since the work of cranking each wheel is equivalent, she just wants to minimize her effort.

To make it a bit harder, she's asked you to solve the problem for a generic apparatus of any setup of wheels. However, she also only cares about full revolutions of the crank, fractional revolutions should not be considered. Also, she doesn't want the possible apparatus to have two wheels of the same radius, so she'll only give you a problem with unique radii. Clearly for any given apparatus it might not be possible to turn the cranks in integer revolutions to exactly traverse her desired distance. In this case Kacey just wants you to tell her it can't be done. She doesn't imagine this is a kind of problem she'll be able to solve quickly on her own. That's why Kacey's come to you to figure out the answer for her.

Input

The input consists of two lines. The first line has two integers, n ($1 \leq n \leq 100$), the number of radii, and t ($1 \leq t \leq 1000$), the target distance. The second line has n integers k_i ($1 \leq k_i \leq t$ for all i), the radii of the wheels.

Output

Output "Possible:" followed by the minimum number of crank revolutions required if it is possible and "Impossible." otherwise.

Sample Input	Sample Output
4 9 2 3 5 10	Possible: 3

Sample Input	Sample Output
2 7 2 4	Impossible.

D: Shrek and Dumbrix

Shrek loves puzzle games. One of his favorite games is Numbrix. In this game, you are given a 9 by 9 starting grid with some positions containing values and some blank. The point of the game is to fill in the blank spots so that there is a continuous chain of numbers between 1 and 81. In other words, all values from 2 to 80 are directly connected to the values 1 greater and 1 less than themselves. Connections are only allowed through single movements on the vertical and horizontal axes (i.e. no diagonal movement). It is guaranteed that each Numbrix grid has only 1 possible solution.

Shrek likes Numbrix, but he thinks it might be too hard for you. He has decided to relax the rules a little bit so you might have a chance of solving the puzzles too. In his revised version of the game, Dumbrix, he provides two additional guarantees. The given board always contains the numbers 1 and 81. Also, for any continuous sequence of values between 1 and 81 of length ≥ 6 at least one value in the sequence exists in the given board.

Input

The input consists of 9 lines each containing 9 integer values. The value at each line i and position j, $v_{i,j}$ ($0 \leq v_{i,j} \leq 81$), represents its corresponding board position's value. Positions with a value of 0 represent unknown values.

Output

Output the solved board with a single space between each value in a row and a new line after each row.

Sample Input

```
1 0 19 0 37 0 55 0 73
0 17 0 35 0 53 0 71 0
3 0 21 0 39 0 57 0 75
0 15 0 33 0 51 0 69 0
5 0 23 0 41 0 59 0 77
0 13 0 31 0 49 0 67 0
7 0 25 0 43 0 61 0 79
0 11 0 29 0 47 0 65 0
9 0 27 0 45 0 63 0 81
```

Sample Output

```
1 18 19 36 37 54 55 72 73
2 17 20 35 38 53 56 71 74
3 16 21 34 39 52 57 70 75
4 15 22 33 40 51 58 69 76
5 14 23 32 41 50 59 68 77
6 13 24 31 42 49 60 67 78
7 12 25 30 43 48 61 66 79
8 11 26 29 44 47 62 65 80
9 10 27 28 45 46 63 64 81
```

Sample Input

```
35 0 0 0 41 0 0 0 47
0 0 0 0 0 0 0 0 0
0 0 29 0 57 0 51 0 0
0 0 0 27 0 0 0 0 0
7 0 9 0 0 0 63 0 67
0 0 0 0 0 0 0 0 0
0 0 13 0 21 0 73 0 0
0 0 0 0 0 0 0 79 0
1 0 0 0 17 0 0 0 81
```

Sample Output

```
35 36 39 40 41 42 43 46 47
34 37 38 55 54 53 44 45 48
33 30 29 56 57 52 51 50 49
32 31 28 27 58 61 62 65 66
7 8 9 26 59 60 63 64 67
6 11 10 25 24 23 72 71 68
5 12 13 20 21 22 73 70 69
4 3 14 19 18 75 74 79 80
1 2 15 16 17 76 77 78 81
```

E: Gas Light

Andrew is driving to Akron for the weekly Friday ACM programming practice when suddenly his gas light comes on. He's not sure whether he has enough gas to make it to practice or not. Given the amount of miles he can drive before he runs out of gas he needs help finding out if he can make it to Akron or not.

Each city has one or more roads to other cities. Each road between cities is exactly 1 mile long. None of the roads are one-way. It is guaranteed to be possible to reach any city from any other city through some series of roads.

Given that Andrew starts on city 1, and there are n cities with the nth city representing Akron, can Andrew make it to Akron before he runs out of gas?

Input

The first line of input will contain 3 integers, n ($1 \leq n \leq 100$), the number of cities, k ($n-1 \leq k \leq \frac{n(n-1)}{2}$), the number of roads, and m ($1 \leq m \leq 20$), the number of miles until Andrew runs out of gas. The next m lines contain the descriptions of the roads. The ith road is described by 2 integers, a_i and b_i ($1 \leq a_i, b_i \leq n$), the two cities connected by the road.

Output

Output "Yes" if Andrew can make it to Akron or "No" if he cannot, followed by a new line.

Sample Input

```
5 4 4
1 2
2 3
3 4
4 5
```

Sample Output

```
Yes
```

Sample Input

```
5 4 3
1 2
2 3
3 4
4 5
```

Sample Output

```
No
```

F: Word Hops

Tina has a curious computer that operates on strings. Her computer is able to move from one space separated word to another instantly. It can also move from one character to another instantly. Tina tells you that she considers a “move” an instance of the computer moving it’s address pointer either by one character or by one word. She also tells you that the computer always starts pointing at the very first character which is the zero address of the programs memory.

Although Tina has managed to get her hands on some interesting hardware, she’s not really a software person. She’s asked you to help her out in writing a program to determine the minimum number of moves required to get to an address. That is, if the computer has the state “Hello World”, with the target address of 7, then it could move the pointer once from the 0 address to the 6 address (now pointing at W), and then once more to the 7 address. That means it would take the computer two moves to get the 7 address with the initial state “Hello World”.

Given the number of words, the target address, and the initial state of the computer, your job is to figure out the minimum number of moves for Tina.

Input

The input consists of two lines. The first line has two integers, w $(1 \leq w \leq 100)$, the number of words and t $(1 \leq t \leq 1000)$, the target index. The second line is a sequence w space separated words with no special characters.

Output

Output the minimum number of moves required to reach the target index and the character at the target index.

Sample Input

```
2 6
Hello World
```

Sample Output

```
1 W
```

Sample Input

```
6 20
This is the best sentence ever
```

Sample Output

```
7 t
```