ACM ICPC

UM Qualification

October 1, 2016

Problems

A	Combined Celery
B	Chessboard Path Sums
C	Chessboard Independent Sums
D	Balls and Bins
E	Super Mario
F	Fighting Queens
G	Box Stacking
H	Happy Party
I	Compository Number Sequence
J	Market Pioneer

Problem A
Combined Celery

Dorothy Grumples is a student at Combined Celery (CC) Company. As with any company, CC has had some very good times as well as some very bad time. Dorothy does trending analysis of the stock prices for CC, and she wants to determine the largest decline in stock prices over various time spans. For example, if over a span of time the stock prices were 19, 12, 13, 11, 20 and 14, then the largest decline would be 8 between the first and fourth price. If the last price had been 10 instead of 14, then the largest decline would have been 10 between the last two prices.

Dorothy has done some previous analyses and has found that the stock price over any period of time can be modeled reasonably accurately with the following equation:

$$price(k) = p * (sin(a * k + b) + cos(c * k + d) + 2)$$

where p, a, b, c and d are constants. Dorothy would like you to write a program to determine the largest price decline over a given sequence of prices. Figure A.1 illustrates the price function for Sample Input 1. You have to consider the prices only for integer values of k.

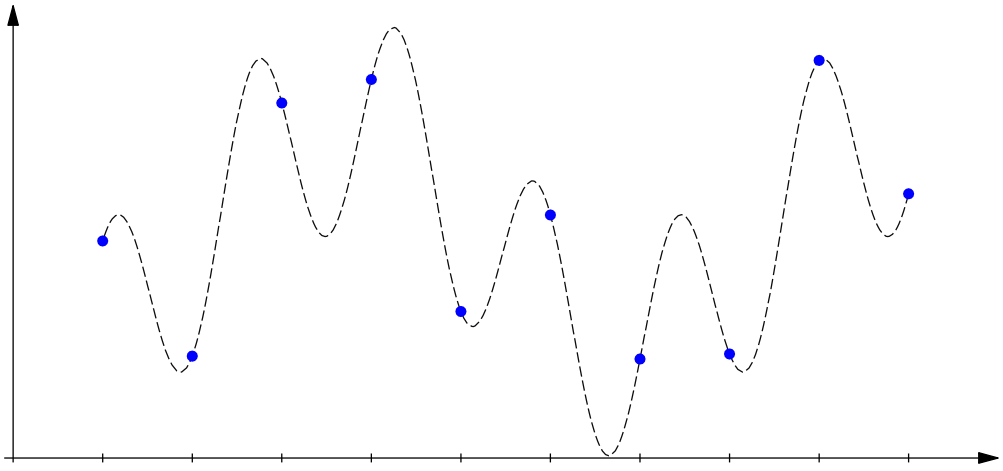


Figure A.1: Sample Input 1. The largest decline occurs from the fourth to the seventh price.

Input

The input consists of a single line containing 6 integers p ($1 \leq p \leq 1000$), a, b, c, d ($0 \leq a, b, c, d \leq 1000$) and n ($1 \leq n \leq 10^6$). The first 5 integers are described above. The sequence of stock prices to consider is price(1), price(2), ... , price(n).

Output

Display the maximum decline in the stock prices. If there is no decline, display the number 0.000000. Your output should have exactly six digits to the right of the decimal point.

Sample Input 1

```
42 1 23 4 8 10
```

Sample Output 1

```
104.855110
```

Sample Input 2

```
100 432 406 867 60 1000
```

Sample Output 2

```
399.303813
```

Problem B
Chessboard Path Sums

Alice and Bob are having a discussion about who is the best problem solver at UM. They decide to determine this by solving each other's toughest problems. Alice decides to go first.

Alice: You are given an $n \times m$ chessboard, where each position contains a positive integer. You are allowed to walk from the top left corner to the bottom right corner once, and at every position, you are only allowed to move one step right, or one step down. We'd like to know the maximum sum possible of all the values touched on an optimum path.

Bob: Hah! This is an easy dynamic programming problem. Let $f[i][j]$ represent the maximum possible sum from the top left corner to position (i, j), then...

Alice: OK, I already know you got it. But what happens if you do the walk twice?

Bob: Then the answer is of course twice as much!

Alice: No, if you visit the same position twice, that value will only be counted once.

Bob: Er... OK, then this is still a dynamic programming problem. Let $f[i][j][k][l]$ represent the maximum possible sum of two paths, both start from the top left corner, and one ends at (i, j), and the other ends at (k, l).

Alice: OK, this is correct. But what if you can go t times?

Bob: We can still use dynamic programming. Let ...

Alice: Wait a second. How large is your array then?

Bob: It is $(n \times m)^t$. Well, I can bring it down to $(n \times m) \times (\min(n, m))^t$.

Alice: But it is still exponential to the input size.

Bob: Yes...

Bob is now asking for your help!

Input

The first line contains three integers, n, m, t $(1 \leq n, m, t \leq 100)$.
The following n lines, each contains m integers. All the integers are positive and smaller than or equal to 1000.

Output

A single integer: the maximum possible sum of t paths.

Sample Input

```
3 3 2
1 2 1
2 3 0
1 0 1
```

Sample Output

```
10
```

Problem C
Chessboard Independent Sums

After spending too much time on the mishap with problem B, Bob finally realized that something was missing!

Bob: This is unfair! You give me your problem, but you don't have to solve my problem!

Alice: OK, then, what is your problem?

Bob: Let's do the chessboard sum again. We still have a chessboard of size $n \times m$, but now you are allowed to select whatever the number of positions you want, and calculate the sum of these values. You want to maximize the sum.

Alice: Then obviously, the solution is to take all the positive numbers.

Bob: Nah, that would be too easy. In my problem, you are not allowed to select two numbers a, b at the same time if these two positions are neighbors.

Alice: You mean these two positions share an edge?

Bob: Yes. For example, $(1, 2)$ and $(1, 3)$ are neighbors, but $(1, 1)$ and $(2, 2)$ are not, because they share a common point, but not an edge. If $(1, 2)$ contains $a_{1,2}$, $(1, 3)$ contains $a_{1,3}$, then you cannot take $a_{1,2}, a_{1,3}$ at the same time. Note that taking $a_{1,2}$ would not affect $a_{1,4}$ at all, even if when $a_{1,3} = a_{1,4}$.

Alice: OK, this problem looks interesting.

Now Alice is a bit stuck. Help Alice solve the problem!

Input

The first line contains two integers: n, m $(1 \leq n, m \leq 100)$.
The following n lines, each contains m integers. All the integers will be between 0 and 1000 inclusively.

Output

A single integer: the maximum possible sum, where no two numbers selected for the sum are adjacent to the other (adjacent here means sharing an edge).

Sample Input

```
3 3
1 2 1
2 3 0
1 0 1
```

Sample Output

```
7
```

Explanation

7 = 1 + 1 + 3 + 1 + 1

None of these numbers are adjacent to each other and they form maximum possible sum.

Problem D
Balls and Bins

We have n bins, and bin i currently has $x[i]$ balls in it. We want to do some operations so that the final configuration is $y[1 \ldots n]$, and also minimize the cost. Here are the three operations we can do.

- Make a new ball, and put it in a bin i. The cost is X.
- Take out a ball from a bin, and destroy it. The cost is Y.
- Take a ball from bin i, and put it in bin j. The cost is $Z \times |i - j|$.

Input

First line contains four integers: n, X, Y, Z ($1 \leq n \leq 200, 0 \leq X, Y, Z \leq 10000$).
Second line contains n integers, $x[1], \ldots, x[n]$, which represents the initial configuration.
Third line contains n integers, $y[1], \ldots, y[n]$, which represents the final configuration.
For all inputs, we have $0 \leq x[i], y[i] \leq 10$.

Output

A single integer represents the minimum cost.

Sample Input

```
4 2 2 1
1 1 1 0
0 0 0 5
```

Sample Output

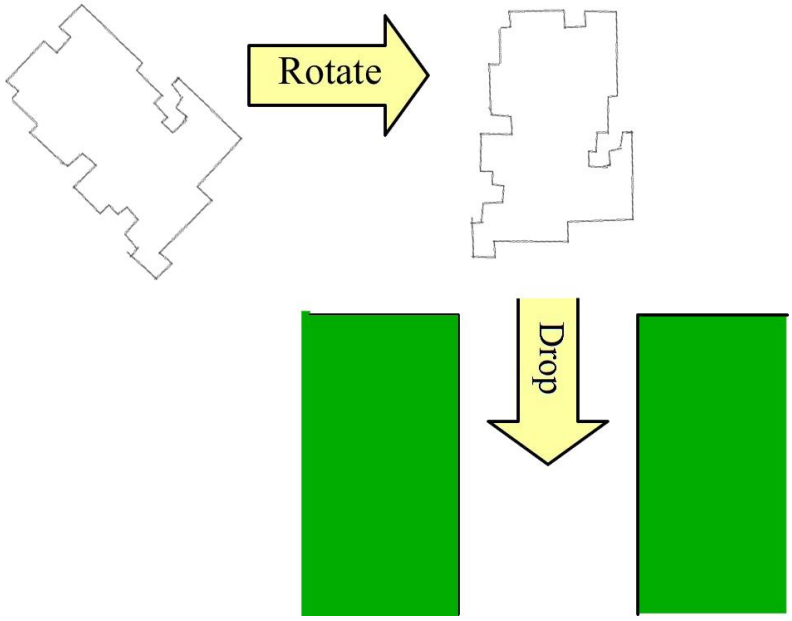
```
10
```

Problem E
Super Mario

You are working on a new awesome Super Mario game clone and everything is going along splendidly, except one little thing… In your game Super Mario exits are marked with pipes of various widths and your game character Mario must be able to fit through the pipe in order to advance to the next level. To keep your screen game space as spacious as possible to populate it with Goombas, you decide to design you pipes to be as small as possible in width, yet still just wide enough to accommodate your Super Mario character.

For your game sprite character, you will use a polygonal approximation, where Super Mario (modeled as a polygon) must fit into the pipe. Before hopping into the pipe, he can be rotated by any arbitrary angle in order to minimize the required width of the pipe. Once inside the pipe he will travel downwards and will not shift or rotate until he reaches the next level.

The following figure shows how Mario is first rotated so that he fits into the pipe.



Your task is to compute the smallest pipe width that will allow a given Mario polygon to pass through the pipe opening and onto the next level.

Input

The input contains several test cases. Each test case starts with a line containing an integer n $(3 \leq n \leq 100)$, the number of points in the polygon that models Super Mario.

The next n lines then contain pairs of integers x_i and $y_i (0 \leq x_i, y_i \leq 10^4)$, giving the coordinates of the polygon vertices in order. All points in one test case are guaranteed to be mutually distinct and the polygon sides will never intersect. (Technically, there is one inevitable exception of two neighboring sides sharing their common vertex. Of course, this is not considered an intersection.)

The last test case is followed by a line containing a single zero.

Output

For each test case, display its case number followed by the smallest pipe width through which Super Mario can escape to the next level. Display the minimum width with exactly two digits to the right of the decimal point, **rounding up** to the nearest multiple of 1/100.

Follow the format of the sample output.

Sample Input

```
3
0 0
3 0
0 4
4
0 10
10 0
20 10
10 20
0
```

Sample Output

```
Case 1: 2.40
Case 2: 14.15
```

Problem F
Fighting Queens

In the game of chess, the queen is a powerful piece. It can attack by moving any number of spaces in its current row, in its column or diagonally.

In the eight queens puzzle, eight queens must be placed on a standard 8×8 chessboard so that no queen can attack another. The center figure below shows an invalid solution; two queens can attack each other diagonally. The figure on the right shows a valid solution. Given a description of a chessboard, your job is to determine whether or not it represents a valid solution to the eight queens puzzle.

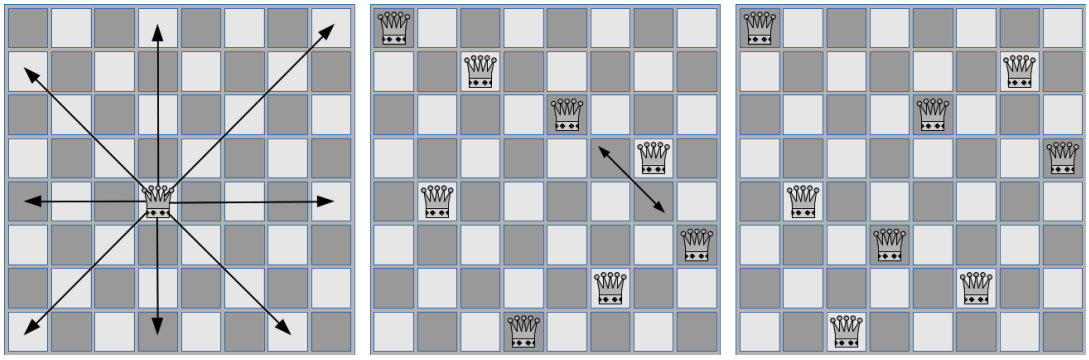


Figure 1: Queen movement (left), invalid solution (center), valid solution (right).

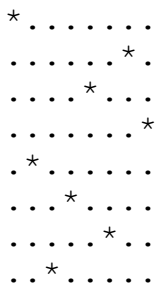
Input

Input will contain a description of a single chessboard, given as eight lines of eight characters each. Input lines will consist of only the characters '.' and '*'. The '.' character represents an empty space on the board, and the '*' character represents a queen.

Output

Print a single line of output. Print the word "valid" if the given chess board is a valid solution to the eight queens problem. Otherwise, print "invalid".

Sample Input



Sample Output

```
valid
```

Problem G
Box Stacking

Description

Leslie sells boxes. All her boxes are rectangular but come in many different sizes. Leslie wants to create a really eye-catching display by stacking, one on top of another, as many boxes as she can outside her store. To maintain neatness and stability, she will always have the sides of the boxes parallel but will never put a box on top of another if the top box sticks out over the bottom one. For example, a box with base 5-by-10 cannot be placed on a box with base 12-by-4.

Of course the boxes have three dimensions and Leslie can orient the boxes anyway she wishes. Thus a 5-by-10-by-12 box may be stacked so the base is 5-by-10, 5-by-12, or 10-by-12.

For example, if Leslie currently has 4 boxes of dimensions 2-2-9, 6-5-5, 1-4-9, and 3-1-1, she could stack up to 3 boxes but not all four. (For example, the third box, the first box, then the last box, appropriately oriented. Alternatively, the second box could replace the third (bottom) box.)

Leslie' stock rotates, so the boxes she stacks outside change frequently. It's just too much for Leslie to figure out and so she has come to you for help. Your job is to find the most boxes Leslie can stack up given her current inventory. Leslie will have no more than 10 different sized boxes and will use at most one box of any size in her display.

Input

A positive integer n ($n \leq 10$) will be on the first input line for each test case. Each of the next n lines will contain three positive integers giving the dimensions of a box. No two boxes will have identical dimensions. None of the dimensions will exceed 20. A line with 0 will follow the last test case.

Output

For each test case, output the maximum number of boxes Leslie can stack using the format given below.

Sample Input

```
4
2 2 9
6 5 5
1 4 9
3 1 1
3
2 4 2
1 5 2
3 4 1
0
```

Sample Output

```
Case 1: 3
Case 2: 3
```

Problem H
Happy Party

A company is going to hold a party, and any employee can attend. The company structure is as such that there is a single big boss, and aside from the boss, each employee has a direct supervisor.

It has been established that each employee has a certain happiness value when they go to the party and that each employee is happiest when they have no direct supervisor present at the party. When the employee and their direct supervisor are together at the party, that employee's happiness is 0.

Find a way to maximize the sum of all employees' happiness values. You can assume that there is a single big boss, and that the company employees form a tree-like structure.

Input

The first line contains an integer n ($1 \leq n \leq 200000$), which is the number of employees in the company. The second line contains n integers $f_i (1 \leq i \leq n)$. The i-th integer represents the direct supervisor of employee i. If for some i, $f_i = -1$, then that employee is the big boss of the company. There is exactly one $f_i = -1$. The third line contains n integers, which represents the happiness of all employees. The happiness is between 0 and 1000.

Output

A single integer represents the maximum happiness.

Sample Input

```
7
-1 1 1 2 2 3 3
2 8 10 5 5 4 4
```

Sample Output

```
20
```

Note: In this case employees #3, 4, 5 can show up (without their direct bosses) thereby maximizing happiness value across all employees.

Problem I
Compository Number Sequence

We shall call a number sequence "compository" when it has a property of having each adjacent pair of integers in the sequence sum to a composite number. Composite number is one that is non-prime.

We want to find such a sequence after rearranging a sequence of *consecutive* integers $n, n+1, n+2, \ldots, m$. For example, if $n = 1$ and $m = 10$, one such compository sequence is 1, 3, 5, 4, 2, 6, 9, 7, 8, 10. This is also the lexicographically first such sequence. We can call this a 2-compository sequence, since each 2 adjacent numbers sum to a composite number.

We can extend this definition by defining a degree d-compository sequence as one where all *consecutive* subsequences of length $2, 3, \ldots, d$ sum to a composite number.

For example, the previously stated sequence satisfies 2-compository definition, but not a 3-compository, since the subsequence 5, 4, 2 sums to 11, which is a non-composite number. The lexicographically-first 3-compository sequence is 1, 3, 5, 4, 6, 2, 10, 8, 7, 9.

Input

Input will consist of multiple input sets. Each set will consist of three integers, $n, m,$ and d on a single line. The values of n, m and d will satisfy $1 \leq n < m \leq 1000$, and $2 \leq d \leq 10$. The line 0 0 0 will indicate end of input and should not be processed.

Output

For each input set, output a single line consisting of a comma-separated list of integers forming a degree d-compository sequence (do not insert any spaces and do not split the output over multiple lines). In the case where more than one such sequence exists, print the lexicographically first one (i.e., output the one with the lowest first value; in case of a tie, the lowest second value, etc.). In the case where no d-compository sequence exists, output

```
No compository sequence exists.
```

Sample Input

```
1 10 2

1 10 3

1 10 5

40 60 7

0 0 0
```

Sample Output

```
1,3,5,4,2,6,9,7,8,10
1,3,5,4,6,2,10,8,7,9
No compository sequence exists.
40,41,43,42,44,46,45,47,48,50,55,53,52,60,56,49,51,59,58,57,54
```

Problem J
Market Pioneer

Richard is a market Pioneer. He dives into a new market called UM16.

Initially, he has F units of funding money. There are a total of M stocks Richard may buy or sell. For ith stock, Richard can buy it for C_i units of money. Later he can sell it, earning E_i units of money.

Note that it is guaranteed that E_i is always strictly smaller than C_i. Richard can buy and sell stocks in any order, but each stock can only be bought and sold once.
As you may have noticed, the best way to earn money is to do nothing!
However, that is not the spirit of the market! Richard can only buy a stock if he has enough money for it. Compute the maximum number of stocks Richard can buy and sell.

Input

The first line contains two integers, F $(1 \leq F \leq 5000)$ and M $(1 \leq M \leq 50)$.
The second line contains M integers. The i-th integer represents money cost by the i-th stock. Each number in this line is between 1 and 5000.
The third line contains M integers. The i-th integer represents money earned by the i-th stock. Each number in this line is between 0 and 5000.

Output

A single integer represents the maximum number of buys and sells that Richard can complete.

Sample Input 1

```
10 1
10
0
```

Sample Output 1

```
1
```

Sample Input 2

```
12 4
4 8 2 1
2 0 0 0
```

Sample Output 2

```
3
```