



Problem A
The Table

Input File: A.IN

Output File: standard output

Program Source File: A.C, A.CPP, A.JAVA

Consider the table of 32-bit signed integers with n rows and m columns. The columns are numbered from 1 to m beginning from the left side of the table. Let A_i ($1 \leq i \leq m$) is the product of all numbers in the i -th column. Find the maximum of these products and print the column number where this maximum product is achieved. If there are many such columns, print the largest number of the column.

Input. Consists of multiple tests. Each test begins with a line with two integers m and n ($1 \leq m \leq 20$, $1 \leq n \leq 1000$). Each of the next n lines contains m 32-bit signed integers.

Output. For each test case print on a separate line the column number with the maximum product. If there are several of them – print the largest number of such column.

Sample Input

```
3 3
20 10 30
15 20 20
30 30 20
3 2
2 -2 2
2 -2 2
```

Sample Output

```
3
3
```




Problem C
Largest Empty Circle on a Segment

Input File: C.IN

Output File: standard output

Program Source File: C.C, C.CPP, C.JAVA

We are given N line segments on the 2D plane. We want to find the maximum radius of an empty circle whose center coordinates (x_c, y_c) are constrained as follows:

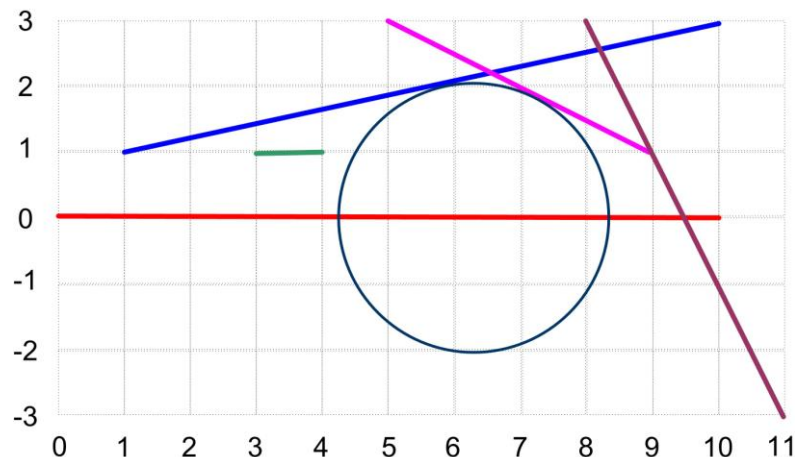
- $0 \leq x_c \leq L$
- $y_c = 0$

A circle is empty if no part of a segment is located strictly inside of it (thus, a segment may touch the circle, but may not intersect with the interior of the circle).

The first line of the input file contains the number of test cases T . The test cases are described next. The first line of a test case contains the integer numbers N and L ($1 \leq N \leq 2000$ and $0 \leq L \leq 10000$). The next N lines of the test case contain 4 integers each, describing the coordinates of the endpoints of a segment: x_a, y_a, x_b and y_b . The coordinates of the endpoints of the segment are (x_a, y_a) and (x_b, y_b) . All the coordinates are between -20000 and +20000. Every two consecutive numbers on the same line are separated by a single blank.

For each test case print a line containing a real number R , denoting the maximum radius of an empty circle whose center obeys the constraints. The number must be printed with 3 decimal digits (the number must be rounded up or down according to the usual rounding rules).

Sample Input	Sample Output
1 4 10 1 1 10 3 5 3 9 1 3 1 4 1 8 3 11 -3	2.118





Southeastern European Regional Programming Contest
Bucharest, Romania
October 16, 2010

Problem D
Problemsetting

Input File: D.IN

Output File: standard output

Program Source File: D.C, D.CPP, D.JAVA

It's well-known that different programming contests require different kind of problems. For example, maximal array size for TopCoder problem is only 50, and you definitely can not give a Suffix Tree problem to IOI because children will not have a chance to solve it (except of some touristic-inclined ones). Thus not every problem is acceptable for every contest.

You are preparing problemsets for N different contests. These contests require different number of problems, depending of type. For example, ACM ICPC style problemset usually has 10 problems, TopCoder SRM – 5 and so on.

Luckily you have already prepared M different problems. For each problem you have determined a set of contests you can give that problem to. Also you know the required number of problems for each contest.

Find out the maximal number of different contests for which you can simultaneously compose *complete* problemsets from the given set of problems. All problems in the problemsets must be unique, i.e. no problem can be used twice in different problemsets.

The input file contains several test cases.

The first line of each test case contains 2 integers N and M ($1 \leq N \leq 15$, $0 \leq M \leq 50$) – the number of different contests and the number of prepared problems. Each of the following N lines contains the name of contest, followed by the required number of problems for that contest. The name of a contest consists of lower- and uppercase Latin letters and/or digits, is not empty and does not exceed 100 characters. Contest names are case-sensitive. It's guaranteed that all contest names will be pairwise different. The required number of problems does not exceed 100.

Each of the following M lines contains a (possibly empty) list of acceptable contest names for each problem, separated by a single space. It's guaranteed that all contest names will be correct (i.e., noted in the previous section of the current test case) and unique.

The line containing two zeroes indicates the end of the input file.

For each test case print an answer for that case on a new line, as shown in the sample output.

Sample input	Sample output
4 5 IOI 3 IPSC 2 TopCoder 2 SEERC 10 IOI IPSC TopCoder IOI IPSC IOI IPSC TopCoder SEERC 1 1 SampleContest 1 SampleContest 0 0	Case #1: 2 Case #2: 1



Problem E
Control Points

Input File: E.IN

Output File: standard output

Program Source File: E.C, E.CPP, E.JAVA

Joe is a skilled programmer. The management of the highway police division wants to control the traffic on the highway, and hired him to solve the following problem. There are some fixed control points on the highway. The highway police division wants to place specialized devices to record information from all the control points. Such a device is very expensive and unfortunately can cover an interval of only one meter (including the left endpoint and excluding the right endpoint). The management wants to place the minimum number of devices to cover all the control points. Joe must find an efficient solution.

The program input is from a text file. Each data set in the file stands for a particular configuration of control points. A data set starts with the number n ($n \leq 100000$) – the number of control points, followed by the real values in ascending order describing the coordinates of the control points (the highway is considered a real line). The coordinates are expressed in meters. White spaces can occur freely in the input. The input data terminate with an end of file.

For each set of data the program has to print the minimum number of devices needed to cover all the control points to the standard output.

An input/output sample is in the table bellow. There are two data sets. In the first case, there are 2 points: 3.56 and 4. Only one device is needed.

Sample Input	Sample Output
2	1
3.56 4	2
3	
1 2 2.9	



Southeastern European Regional Programming Contest
Bucharest, Romania
October 16, 2010

Problem F
Gamers

Input File: F.IN

Output File: standard output

Program Source File: F.C, F.CPP, F.JAVA

As you probably know, the streets and crossroads of Bucharest form a perfect grid. What you probably don't know, is that there is exactly one gaming club and exactly one gamer on each of these crossroads. Strangely enough, each gaming club offers exactly one game. Gamers are also a bit strange, but they generally live a simple life, determined by the following rules:

1. A gamer never plays in the club of his own crossroads. Never!!!
2. During a day each gamer has to play each of the games when this is not contradicting the Rule 1.
3. Gamers travel from one crossroads to another only by horizontals or verticals.
4. A gamer can not go directly from one club to another. It has to come back home and eat something first and has to finish the day in his own intersection.
5. A gamer has to live optimally, thus always picking the best possible strategy to implement the above rules, so that she has enough time for gaming.

Having in mind that all the gamers are the same, the Association of Computer Maniacs decided that the city should be optimized in order to minimize the total non-gaming effort. The *non-gaming effort* is the number of crossroads that a gamer walks through the day. The *total non-gaming effort* is the sum of the non-gaming efforts for all gamers in the city. In order to perform the optimizations, ACM needs a program that calculates the total non-gaming effort for a given description of the city.

Several city descriptions are given in the standard input. Each of them starts with a line of two integers R and C ($1 \leq R, C \leq 1000$), denoting the number of rows and the number of columns of the crossroads in the city. R lines of C characters follow, denoting the kind of game that is offered by the club of each of the crossroads. Each game is coded using a single digit ('0' to '9').

For each of the city descriptions the program has to write on a separate line of the standard output a line containing one integer – the total non-gaming effort for the city.

Sample input	Sample output	Explanation
2 2 12 12 3 3 122 313 321	16 72	In first case, each of the four gamers has to play games 1 and 2. Hopefully, both are only one intersection away, so the non-gaming effort for each gamer is 4 (from home to club 1, from club 1 to home, from home to club 2 and from club 2 to home). That means a total non-gaming effort of 16. In the second case, each of the 9 gamers has two of the games 1 intersection away, and one – 2 intersections away, meaning total non-gaming effort of $9 \cdot (1+1+1+1+2+2) = 72$.



Problem G

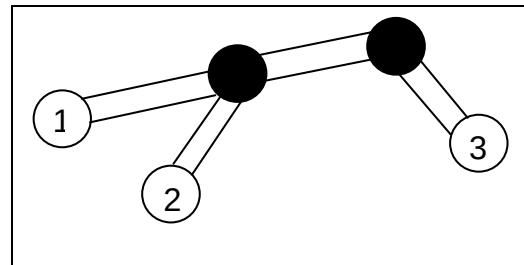
Cosmic Station

Input File: G.IN

Output File: standard output

Program Source File: G.C, G.CPP, G.JAVA

The young programmers Peter and Stancu were hired by two cosmic agencies. The agency of Peter constructed a station, composed of modules. Some couples of modules are linked by corridors in such way that it is possible to go from each module to each other module by unique path of corridors. There is neither a corridor that starts and finishes in the same module nor two or more corridors that link same two modules. *Outside* modules are linked to exactly one other module (white on the figure) and are labeled from 1 to N . They are just for fun. All equipment of the station is concentrated in the *inner* modules – linked to more than one other module (black on the figure). The chiefs of Peter would like to keep in secret the number of inner modules. In order to hide the number of inner modules Peter encoded the topology of the station by giving for each two outside modules the distance between them (i.e. the number of corridors on the unique path between the modules).



Now, Stancu has a difficult task – he promised to his bosses to decrypt the coding of Peter and to find the number of inner modules of the station. Unfortunately, he is not experienced enough. Help him. Write a program to solve the task.

On the first line of the input file the number T of the test cases will be given. Each test case starts with a line containing the number N of the outside modules ($3 \leq N \leq 1024$). Then $N - 1$ lines follow. On the first of these lines the distances from outside module 1 to the outside modules 2, 3, ..., N are given, separated by single spaces. On the second line are given, also separated by single spaces, the distances from outside module 2 to the outside modules 3, 4, ..., N , and so on. The last line contains only the distance from the outside module $N - 1$ to the outside module N .

For each test case the program has to print on a separate line of the standard output the number M of the inner modules of the station. In all test cases M will be less than 1024.

Sample input	Sample output
1 3 2 3 3	2



Problem H

Game

Input File: H.IN

Output File: standard output

Program Source File: H.C, H.CPP, H.JAVA

Bill is fond of computer games. He likes to analyze games and to provide efficient solutions. Now, he is studying the following game. The game starts with a $n \times n$ matrix filled with positive integers. When it is her/his turn, a player can delete the last row or the last column of the matrix, if the sum of the numbers in that row/column is even. If a player cannot delete the last row or the last column on his turn, then he loses the game. Bill thinks that this game can be classified as first player wins (W) or first player loses (L). First player wins means that the first player has a strategy to win, no matter how the second player plays the game. First player loses means that no matter what the first player does the second player has a strategy to win.

Bill is also a skilled programmer. He wants to write a program to classify the game quickly. Can you help him?

The program input is from a text file. Each data set in the file stands for a particular game. A data set starts with the number n ($n \leq 1000$), the matrix dimension, followed by the positive integers in the matrix. The program has to print W if the first player wins the game, or L if the first player loses the game.

White spaces can occur freely in the input. The input data are correct and terminate with an end of file. For each set of data the program prints the result to the standard output from the beginning of the line. An input/output sample is in the table bellow. There are two data sets. In the first case, the matrix dimension n is 2. The integers in the matrix are: 2 4 6 8. The result for the data set is L , meaning that no matter what the first player does, the second player wins.

Sample Input	Sample Output
2	L
2 4	W
6 8	
3	
5 4 2	
1 5 9	
7 3 8	



Problem I Proof Generator

Input File: I.IN

Output File: ordered output

Program Source File: I.C or I.CPP or I.JAVA

A logical formula has the syntax shown in figure 1(a), where a variable stands for a truth value, the formula $(+ F_1 \dots F_n)$ stands for the logical disjunction of the formulae F_i , $(* F_1 \dots F_n)$ denotes the logical conjunction of the formulae F_i , and $\sim F$ is the negation of F . If a formula has the particular syntax given in figure 1(b) we say that the formula is in the ACM Normal Form (ACMNF).

<formula> ::= <variable> | $\sim$ <formula> | $(+<formulae>)$ | $(*<formulae>)$
<variable> ::= a lower case letter from the English alphabet
<formulae> ::= <formula> | <formula><formulae>

a) The general syntax of a formula

<ACMNF_formula> ::= <term> | $(+<terms>)$
<term> ::= <literal> | $(*<literals>)$
<terms> ::= <term><term> | <term><terms>
<literal> ::= <variable> | $\sim$ <variable>
<literals> ::= <literal><literal> | <literal><literals>
<variable> ::= a lower case letter from the English alphabet

b) The ACMNF syntax of a formula

Figure 1. Formula syntax

A formula is converted to ACMNF using the rewriting rules given below, where F represents a formula, s stands for a non empty sequence of formulae, and s and s' denote possibly empty sequences of formulae. Applying a rewriting rule $q \rightarrow r$ on a formula F means to substitute by r a part of F that matches the pattern q , as in shown figure 2. The conversion terminates when no rewriting rule can be applied. The conversion terminates for any formula, and the result is unique regardless which rules are applied on which parts of the formula and in which order.

1. $\sim \sim F \rightarrow F$
2. $\sim (*FS) \rightarrow (+ \sim F \sim (*S))$
3. $\sim (+FS) \rightarrow (* \sim F \sim (+S))$
4. $(+F) \rightarrow F$
5. $(+s (+S) s') \rightarrow (+s S s')$
6. $(*F) \rightarrow F$
7. $(*s (*S) s') \rightarrow (*s S s')$
8. $(*s (+FS) s') \rightarrow (+ (*s F s') (*s (+S) s'))$

$(+ (* (+ \sim (*ab)) (+ \sim a)) c)$	$\rightarrow 4 \rightarrow$
$(+ (* (+ \sim (*ab)) \sim a) c)$	$\rightarrow 2 \rightarrow$
$(+ (* (+ (+ \sim a \sim (*b)) \sim a) c)$	$\rightarrow 6 \rightarrow$
$(+ (* (+ (+ \sim a \sim b)) \sim a) c)$	$\rightarrow 4 \text{ or } 5 \rightarrow$
$(+ (* (+ \sim a \sim b) \sim a) c)$	$\rightarrow 8 \rightarrow$
$(+ (+ (* \sim a \sim a) (* (+ \sim b) \sim a)) c)$	$\rightarrow 4 \rightarrow$
$(+ (+ (* \sim a \sim a) (* \sim b \sim a)) c)$	$\rightarrow 5 \rightarrow$
$(+ (* \sim a \sim a) (* \sim b \sim a) c)$	

Figure 2. Converting a formula to ACMNF

A set of axioms is represented as a list $(v_1 v_2 \dots v_n)$ of variables that are true. A variable that is not in the list is false. A proof of a formula F according to a set of axioms A is a term from the ACMNF of F such that the term is true according to A . For instance, the terms $(*\sim a \sim a)$ and c are the proofs of the formula $(+ (* (+ \sim (*ab)) (+ \sim a)) c)$ according to the axioms (bc) .

The problem is to code a proof generator that for a given formula F , a set of axioms A , and a number k outputs the next k proofs of F in the order in which they appear in the ACMNF of F . If the proofs are exhausted, the generator continues from the first proof of F . For example, generating the first proof of the formula $(+ (* (+ \sim (*ab)) (+ \sim a)) c)$ according to the axioms (bc) yields $(*\sim a \sim a)$. Generating three more proofs produces c , $(*\sim a \sim a)$, and c . If the ACMNF of a formula contains similar terms, as in the last example in figure 3, these terms are considered distinct.

Write a proof generator that reads sets of data from the standard input. The content of a data set is $F A k_1 \dots k_n 0$, $n > 0$, where F is a formula, A is a set of axioms, and $k_1 \dots k_n$ are long integers different than 0. For each k_i , $i=1, n$, the program generates the next $|k_i|$ proofs of F and, if $k_i > 0$, prints these proofs on the standard output. Each printed proof starts from the beginning of a line and there are no white spaces between the characters of the proof. The generated proofs are not printed if $k_i < 0$. White spaces are used freely in the input. A formula has at most 500 characters and a ACMNF term is at most 80 characters long, not counting white spaces. The input data terminate with an end of file, and are correct.

Input	Output
$(+ (* (+ \sim (*ab)) (+ \sim a)) c)$	c
$(bc) \quad -3 \ 1 \ 1 \ 0$	$(*\sim a \sim a)$
$(+\sim x \sim y \sim y) \quad () \quad -2 \ 1 \ 0$	$\sim y$

Figure 3. Input/output sample



Southeastern European Regional Programming Contest
Bucharest, Romania
October 16, 2010

Problem J
Shares

Input File: J.IN

Output File: standard output

Program Source File: J.C, J.CPP, J.JAVA

A group of n persons and the ACM Chief Judge share equally a number of s shares (not necessary all of them). Let x be the number of shares acquired by each person (x must be an integer). The problem is to compute the maximum value of x .

Write a program that reads pairs of integer numbers from an input text file. Each pair contains the values of $1 \leq n \leq 10000$ and $1 \leq s \leq 10^9$ in that order. The input data are separated freely by white spaces, are correct, and terminate with an end of file. For each pair of numbers the program computes the maximum value of x and prints that value on the standard output from the beginning of a line, as shown in the example below.

Sample Input	Sample Output
1 100	50
2 7	2
10 9	0
10 10	0