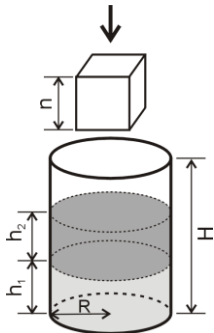


Cocktail

Task code: COCKTAIL

Time limit: 1 sec

Cylindrical vessel of height H ($0 < H \leq 10\,000$) and bottom radius R ($0 < R \leq 10\,000$) stands on a horizontal surface. There are two liquids in the vessel. The liquids have different density and do not mix. D_1 and D_2 are densities of the first and the second liquids respectively, $0 < D_2 < D_1 \leq 10\,000$. h_1 and h_2 are heights of layers of the first and the second liquids, $h_1, h_2 > 0$, $h_1 + h_2 \leq H$. There is a cube made of a solid material. The cube's edge length is n ($0 < n \leq 1.4 \cdot R$), the material's density is D ($0 < D \leq 10\,000$). The cube is put down into the vessel in such a way that one of its sides is horizontal. Pay attention that some amount of liquid may pour out of the vessel. As a result of the operation the cube floats in liquids or lie on the vessel bottom.



You need to calculate the height of the vessel which is occupied by the liquids after the cube is put down into the vessel. You may neglect damping and some other physical effects, but you should take into account Archimedean Principle.

Input data are in the text file COCKTAIL.IN. The only line of the file contains numbers $H, R, D_1, h_1, D_2, h_2, D$ and n separated by spaces. All the numbers are real. Each of them does not contain more than three digits in its fractional part.

Output data are put into the text file COCKTAIL.OUT and contain one real number — the height you need to calculate. Absolute error must not exceed 10^{-3} .

<u>Input data example</u>	<u>Output data example</u>
50 5 2.5 10 1.5 10 1.5 3	20.344

Increasing subsequences

Task code: INCR

Time limit: 4 sec

A sequence $p(1), p(2), \dots, p(N)$ consisting of numbers $1, 2, \dots, N$ is called a *permutation* if all elements in the sequence are different.

It is said that a permutation p contains *increasing subsequence* of k elements when there are numbers $1 \leq i_1 < i_2 < \dots < i_k \leq N$ such that $p(i_1) < p(i_2) < \dots < p(i_k)$.

When a permutation p contains an *increasing subsequence* consisting of B elements and does not contain an *increasing subsequence* consisting of $B+1$ elements then the number B is called the *degree of increase* of this permutation.

You need to write a program which being given a number N calculates the number of permutations whose *degree of increase* is B . Since the number of such permutations might be quite big, it is necessary to calculate its remainder of integer division by 1 000 000 000.

The input file INCR.IN consists of one line. The line contains two integer numbers N and B ($1 \leq N \leq 40$, $1 \leq B \leq 5$) separated by one or more spaces.

The output file INCR.OUT contains one integer number which is the remainder of integer division by 1 000 000 000 of the number of permutations whose *degree of increase* is B .

<u>Input data example</u>	<u>Output data example</u>
3 2	4

Map Generator

Task code: GENMAP

Time limit: 1 sec

The scene of the new computer game “Battles in Space: Unification” (*BSU*) is laid in far future. Mankind is scattered on N planets hostile to each other. A player of the game is a leader of one the planets. Using diplomatic tricks and military the player needs to occupy the rest of the planets and, doing so, reunite mankind.

Special *hyperspace tunnels* are used to travel from one planet to another. Each tunnel connects two of the planets and can be used for communication in both directions. Two planets **can not** be connected by more than one tunnel. The set of all tunnels connecting the planets is called the *map of the game*.

Currently, developers of *BSU* are working on generator of maps of the game. It has been decided to use the special algorithm called «Fine Artificial Model of Intelligence» (*FAMI*) for this purpose. The algorithm operates in the following way. For each pair of planets i и j ($1 \leq i < j \leq N$) a random real number X_{ij} ($0 \leq X_{ij} \leq 1$) is generated. Uniform distribution is used to generate random numbers and numbers X_{ij} are generated independently. Next, if $X_{ij} \leq P$, where parameter P is a real number, then the hyperspace tunnel connecting planets i and j is added into the map being generated.

Developers wish the map generator produced *connected* maps. A map is said to be *connected* when between any pair of planets there is a path, consisting of one or more hyperspace tunnels. Unfortunately, it appears that maps generated by *FAMI* algorithm sometimes are not *connected*. You are assigned to research this phenomenon.

You need to write a program which being given numbers N and P calculates probability of generating *connected* map by *FAMI* algorithm.

Input file GENMAP.IN consists of two lines. The first line contains an integer number N ($1 \leq N \leq 20$), while the second line contains a real number P ($0 \leq P \leq 1$).

Output file GENMAP.OUT needs to contain an answer — the single real number which is the probability of generating *connected* map by *FAMI* algorithm. Absolute error of the answer must not exceed 10^{-2} .

<u>Input data example</u>	<u>Output data example</u>
3 0.5	0.5

Map Generator returns (MG-II)

Task code:
GENMAP2

Time limit: 1 sec

You have just finished your research of *FAMI* algorithm (see task “Map Generator”). You are very proud of yourself and even expect being paid bonus this month. Your boss, whose nickname is Dean, asks you to come over to his office for a moment. You expect thanks and even promotion. But ...

‘What is this?’ asks Dean and shows you your latest report.

‘Well, h-m-m, these are results of my *FAMI* algorithm research, a-a-a,’ you answer. It seems that your dreams about soon promotion are ... just dreams. But you still do not understand what is wrong about your report.

‘I can read, by the way,’ continues Dean. ‘I am talking about the absolute error. Why it is so big? I require better results!’

When you argue with your boss, the best argument is silence. So, now instead of bonuses and promotions you need to rewrite your program.

Input file GENMAP2.IN, as in the previous task, consists of two lines. The first line contains an integer number N ($1 \leq N \leq 20$), while the second line contains a real number P ($0 \leq P \leq 1$).

Output file GENMAP2.OUT needs to contain an answer — the single real number which is the probability of generating *connected* map by *FAMI* algorithm. Absolute error of the answer must not exceed 10^{-8} .

Input data example	Output data example	Input data example	Output data example
3 0.5	0.5	2 1 2	49 1681

Trigonometric optimization

Task code: OPTIMIZE

Time limit: 2 sec

Many problems arising in practical applications may be stated as *optimization problems*. Usually it is necessary to maximize or minimize so called *criterion function* taking into account some *constraints*.

Let's consider a *trigonometric optimization* problem. It is necessary to maximize or to minimize criterion function $F_1(x) + F_2(y) + F_3(z)$ with constraint $x + y + z = S$, where x, y, z – variables, S – parameter, x, y, z, S – natural numbers. Each of the functions F_1, F_2 and F_3 is a trigonometric function *sin* or *cos*.

You need to write a program which solves the *trigonometric optimization* problem.

Input data are in the text file OPTIMIZE.IN, which contains 5 lines. The first line describes function F_1 and contains either *sin* or *cos*. The second and the third lines describe functions F_2 and F_3 respectively and have the same format as the first line. Next, the fourth line contains either *min* or *max*. If the line contains *min* than it is necessary to minimize *criterion function*, otherwise it is necessary to maximize *criterion function*. Finally, the fifth line contains parameter S value ($3 \leq S \leq 1\,000\,000$).

Output data are put into the text file OPTIMIZE.OUT. The only line of the file contains one real number – the found value of the *criterion function*, described in the input file. Absolute error of your answer must not exceed 10^{-10} .

Input data example	Output data example
sin cos sin max 10	2.7787651403

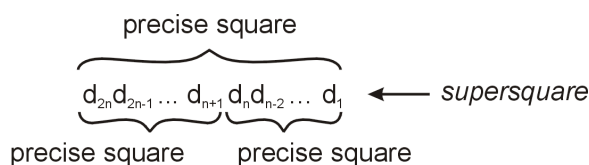
Supersquare

Task code: SUPER

Time limit: 1 sec

Let number A be a *precise square* if there exists natural number B such that $B \cdot B = A$.

$2n$ -digit number without leading zeroes is called a *supersquare* if it is a *precise square* and both n -digit numbers which are formed from its n first digits and its n last digits are *precise squares*. The second n -digit number $d_n d_{n-1} \dots d_1$ formed from n last digits may have leading zeroes but must not be equal zero.



You need to write a program which constructs a $2n$ -digit number which is *supersquare*.

Input file SUPER.IN contains several test cases. The first line contains the number of test cases T ($1 \leq T \leq 10$). Each of the next T lines describes one test case and contains an integer number n ($1 \leq n \leq 500$).

Output file SUPER.OUT consists of T lines, one line per each test case. Each line contains $2n$ -digit *supersquare* number. If several solutions are possible only one of them should be given. When it is impossible to construct $2n$ -digit *supersquare*, the line must contain **NO SUPERSQUARE POSSIBLE** phrase.

Pseudographical recognizer

Task code: PSEUDO

Time limit: 1 sec

Let us define a *pseudographical image* to be a rectangular matrix of the characters '.', '-', '|', '\', and '/'.

The character '.' denotes empty space on the image. A horizontal line segment is given as a set of '-' characters in adjacent cells in the same row of the matrix. A vertical line segment is given as a set of '|' characters in adjacent cells in the same column of the matrix. Similarly, a diagonal line segment is given as a set of '/' or '\' characters in adjacent cells in the same diagonal of the matrix. Of course, a line segment going from Northwest to Southeast has to be given using the '\' characters and a line segment going from Southwest to Northeast using the '/' characters.

Write a program that, given a pseudographical image, determines if it contains exactly one line segment – horizontal, vertical, or diagonal.

Input file PSEUDO.IN contains several test cases. The first line contains the number of test cases T ($1 \leq T \leq 100$). Next follow the descriptions of each test case. The first line of the test case description contains two integers N and M ($1 \leq N, M \leq 10$), the number of rows and columns of the matrix, respectively. Each of the following N lines of the description contains exactly M symbols '.', '-', '|', '\', or '/'.

Output file PSEUDO.OUT consists of T lines, one line per each test case. This line should contain the word **CORRECT** if the input image contains exactly one line segment, or the word **INCORRECT** otherwise.

Input data example	Output data example
5 5 5 \.... .\.... ..\.... 3 3 /.. ./.. ../.. 3 6 3 3 1 1 /	CORRECT INCORRECT INCORRECT INCORRECT CORRECT

"Roman" corridor

Task code: ROME

Time limit: 1 sec

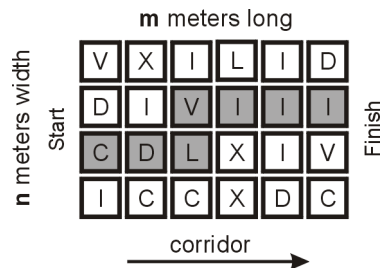
Let's remind the notation of Roman numerals. The notation is for natural numbers from 1 to 3999. Capital Latin letters 'I', 'V', 'X', 'L', 'C', 'D', 'M' and their combinations are used to represent so called *atomic numbers* (see the table below).

1 – I	40 – XL	500 – D
4 – IV	50 – L	900 – CM
5 – V	90 – XC	1000 – M
9 – IX	100 – C	
10 – X	400 – CD	

To put down a number N it is necessary to find the greatest *atomic* number K which is not greater than N . The Roman notation of the found number K is put down, and the process is repeated for $(N-K)$.

The Roman numerals are put down from left to right without spaces. Thus, the number 999 in the Roman notation is **CMXCIX** (but not **IM**, as somebody may think).

You need to pass through a rectangular corridor. The corridor is n meters width and m meters long ($1 \leq n, m \leq 15, n \cdot m \leq 100$). It is laid out by square tiles. Each tile is 1 meter width and has a 'Roman' symbol on it: 'I', 'V', 'X', 'L', 'C', 'D' or 'M'. When passing the corridor, you move from one tile to another. From the current tile you may only move to one of adjacent tiles, vertically or horizontally (but not across). You start at the left and end at the right (see the picture below).



Can you pass through the corridor so that the sequence of symbols on the tiles composing your path was a correct number in the Roman notation? Among all possible solutions you need to find the minimal number.

The input data are in the text file ROME.IN. The first line contains numbers n and m , separated by one or more spaces. Each of the next n lines consists of m characters describing tiles.

The output file ROME.OUT contains one line with the found Roman number or the word **NO** if it is impossible to pass through the corridor in the required way.

Input data example

```
4 6
VXILID
DIVIII
CDLXIV
ICCXDC
```

Output data example

```
CDLVIII
```

Parabolic teleports

Task code: PARABOLA

Time limit: 2 sec

Flatland is a plane with a Cartesian coordinate system O_{xy} . Citizens of Flatland are points that move with the speed 1. Consequently, the minimal time it takes for a Flatlander to move from point V to point W is equal to the length of the line segment VW . This fact is taught in the schools of Flatland as the "shortest path theorem".

However, since George Edward Nius invented the devices called *parabolic teleports*, the theorem no longer holds. A parabolic teleport is a contiguous section of a parabola along which it is possible to move with infinite speed. In other words, it is possible to move from any point on the parabolic teleport to any other point on the same parabolic teleport in zero time.

The points (x, y) that belong to the teleport are given by the formulae $y = A \cdot x^2 + B \cdot x + C$, $XL \leq x \leq XR$, where A, B, C, XL, XR are the parameters of the teleport.

After inventing the teleports, G.E.Nius founded a company to build them. At the moment, the company has already built N of those devices. One might imagine that they are shoveling money left and right, but...

But in practice, the Flatlanders still think the shortest path theorem is true and nobody has even tried the parabolic teleports. To help the situation, G.E.Nius has hired you to write a program that, given

two points V and W , would compute the time it would take to move from V to W using the teleports the company has already built. The idea is that the users of the program will see that the time is less than the length of the segment VW and start to use the invention of Mr. Nius.

Are you up to the task?

Input data are in the text file PARABOLA.IN. On the first line of the file is the integer N ($0 \leq N \leq 100$), the number of the teleports. On the second line of the file are the integers X_V and Y_V ($-100 \leq X_V \leq 100, -1\,000\,000 \leq Y_V \leq 1\,000\,000$), the coordinates of the source point V . On the third line of the file are the integers X_W and Y_W ($-100 \leq X_W \leq 100, -1\,000\,000 \leq Y_W \leq 1\,000\,000$), the coordinates of the destination point W .

Each of the N following lines contains 5 integers, separated by spaces, that describe the i^{th} teleport – giving its parameters $A_i, B_i, C_i, XL_i, XR_i$ ($-100 \leq A_i, B_i, C_i \leq 100, A_i \neq 0, -100 \leq XL_i < XR_i \leq 100$).

Output data should be put to the text file PARABOLA.OUT. The first and only line of the file should contain one real number, the minimal time it takes to move from point A to point B . The absolute error of the answer must not exceed 10^{-4} .

Example of input data

```
2
0 10
0 -10
1 0 0 -10 10
-1 0 0 -10 10
```

Example of output data

```
6.2450
```

Magic pyramid

Task code: PYRAMID

There are many modifications of the famous Rubic's Cube puzzle. One of those, called the Magic Pyramid, is based on a tetrahedron instead of the original cube. Faces of the tetrahedron are labeled with the Latin letters X, U, V , and W . Each face is divided into 9 equilateral triangles numbered from 1 to 36, as shown on Figure 1. To assemble the tetrahedron, the big triangle has to be folded down along the internal thick lines and the matching halves of the external thick lines have to be glued together.

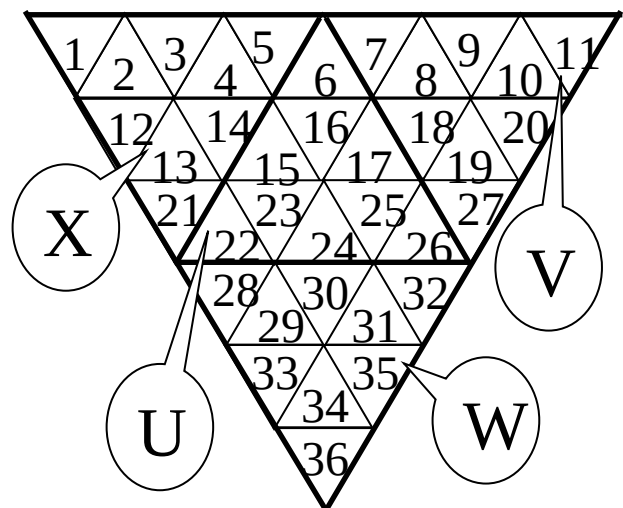


Fig. 1

Each of the small triangles is painted into one of four distinct colors that we label with the Latin letters a, b, c , and d . There are exactly 9 triangles of each of the four colors.

We can transform the tetrahedron using the operation of "rotate a face". To perform this operation, we choose a face of the pyramid and orient the pyramid so that the chosen face is towards us. Then we determine the direction of the rotation (clockwise or counter-

clockwise). Finally, we turn the part of the tetrahedron closest to the chosen face and having the thickness of $1/3$ of the height of the tetrahedron by 120 degrees in the chosen direction, leaving the remaining part of the tetrahedron unmoved. Figure 2 illustrates the effect of rotating the face **U** in clockwise direction.

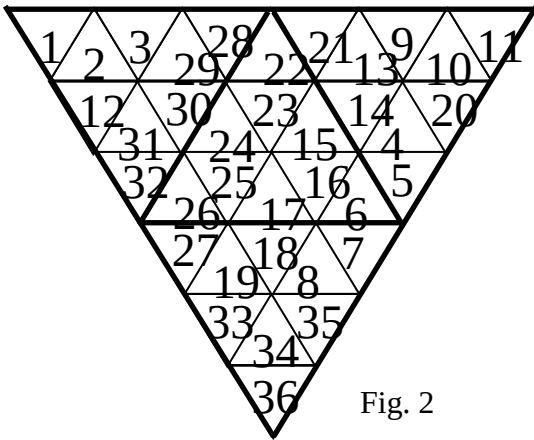


Fig. 2

In addition to rotating the faces, we can also apply the operation of "magic re-paint". This operation is defined by a permutation ($p_1, \dots, p_{36}$) of the integers 1 to 36. The effect of the operation is that the triangle currently in position i (according to the numbering given on Figure 1) is re-painted into the color of the triangle currently in position p_i (according to the same numbering). All 36 triangles are re-painted simultaneously.

The puzzle is solved if, after applying a number of operations "rotate a face" and/or "magic re-paint" all triangles on each face of the tetrahedron have the same color.

Write a program to solve the Magic Pyramid puzzle using the minimal possible number of moves. You may assume that all the test cases are solvable in 9 or less moves.

Input data are in the text file PYRAMID.IN. The first line of the file contains 36 letters 'a' ... 'd'. The letter on the position i denotes the initial color of the triangle number i . It is known that each of the four letters occurs exactly 9 times on the first line of the file.

The second line of the file contains 36 distinct integers 1...36 that describe the "magic re-paint" operation. The i^{th} integer on this line is the member p_i of the permutation that defines the operation.

Output data should be put to the text file PYRAMID.OUT. The first and only line of the file should contain a string of characters 'X', 'U', 'V', 'W', 'x', 'u', 'v', 'w', and '*'. The i^{th} character should describe the i^{th} move in the solution of the puzzle. Each operation "rotate a face" is given as the label of the corresponding face, with lowercase letter denoting a turn in the clockwise and uppercase letter a turn in the counter-clockwise direction. The operation "magic re-paint" is given as the character '*'.

If there are several solutions with the minimal number of moves, output any one of them. If the puzzle can be solved in no moves at all, output an empty line into the file.

Example of input data
 (the second line is wrapped for better layout)

```

ddccccccabbbbbbcaabbbddacccaabddaaa
36 35 34 33 32 31 30 29 28 27 26 25
24 23 22 21 20 19 18 17 16 15 14 13
12 11 10 9 8 7 6 5 4 3 2 1
    
```

Example of output data

*U

Explanation of the example: Figure 3 shows the initial coloring of the tetrahedron, and Figures 4 and 5 show the results of each operation. The colors of the triangles are given by the following patterns:

	a
	b
	c
	d

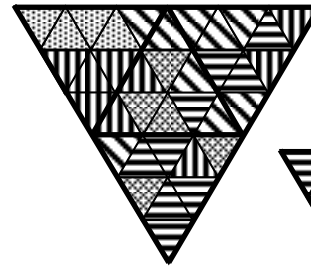


Fig. 3

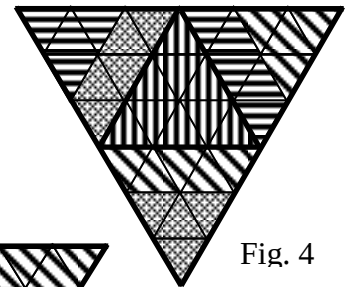


Fig. 4

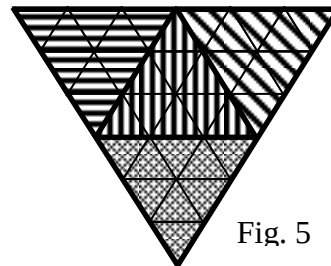


Fig. 5