Problem A
Sum of Consecutive Prime Numbers
Input: A.txt

Some positive integers can be represented by a sum of one or more consecutive prime numbers. How many such representations does a given positive integer have? For example, the integer 53 has two representations $5 + 7 + 11 + 13 + 17$ and 53. The integer 41 has three representations $2+3+5+7+11+13$, $11+13+17$, and 41. The integer 3 has only one representation, which is 3. The integer 20 has no such representations. Note that summands must be consecutive prime numbers, so neither $7 + 13$ nor $3 + 5 + 5 + 7$ is a valid representation for the integer 20.

Your mission is to write a program that reports the number of representations for the given positive integer.

Input

The input is a sequence of positive integers each in a separate line. The integers are between 2 and 10 000, inclusive. The end of the input is indicated by a zero.

Output

The output should be composed of lines each corresponding to an input line except the last zero. An output line includes the number of representations for the input integer as the sum of one or more consecutive prime numbers. No other characters should be inserted in the output.

Sample Input

```
2
3
17
41
20
666
12
53
0
```

Output for the Sample Input

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

Problem B
Book Replacement
Input: B.txt

The deadline of Prof. Hachioji's assignment is tomorrow. To complete the task, students have to copy pages of many reference books in the library.

All the reference books are in a storeroom and only the librarian is allowed to enter it. To obtain a copy of a reference book's page, a student should ask the librarian to make it. The librarian brings books out of the storeroom and makes page copies according to the requests. The overall situation is shown in Figure 1.

Students queue up in front of the counter. Only a single book can be requested at a time. If a student has more requests, the student goes to the end of the queue after the request has been served.

In the storeroom, there are m desks $D_1, \cdots, D_m$, and a shelf. They are placed in a line in this order, from the door to the back of the room. Up to c books can be put on each of the desks. If a student requests a book, the librarian enters the storeroom and looks for it on $D_1, \ldots, D_m$ in this order, and then on the shelf. After finding the book, the librarian takes it and gives a copy of a page to the student.

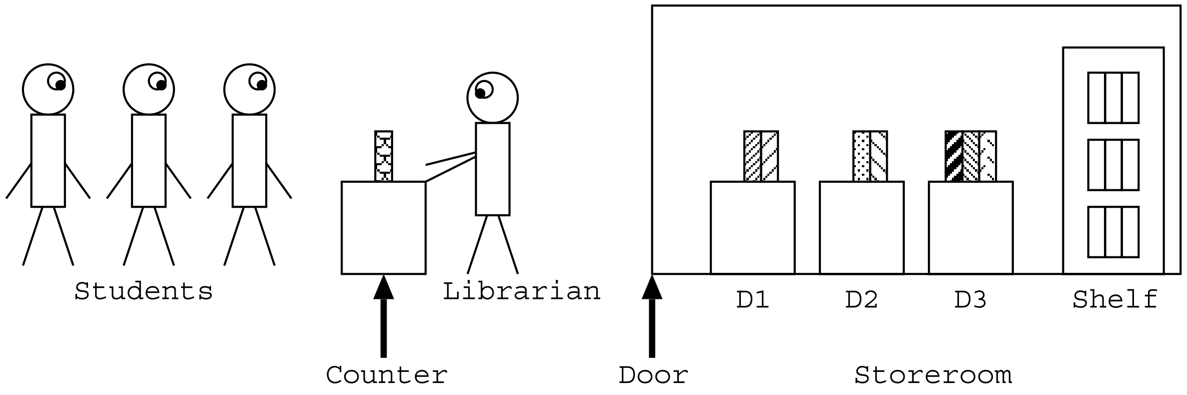


Figure 1: The Library

Then the librarian returns to the storeroom with the requested book, to put it on D_1 according to the following procedure.

- If D_1 is not full (in other words, the number of books on $D_1 < c$), the librarian puts the requested book there.

- If D_1 is full, the librarian
  - temporarily puts the requested book on the non-full desk closest to the entrance or, in case all the desks are full, on the shelf,
  - finds the book on D_1 that has not been requested for the longest time (i.e. the least recently used book) and takes it,
  - puts it on the non-full desk (except D_1) closest to the entrance or, in case all the desks except D_1 are full, on the shelf,
  - takes the requested book from the temporary place,
  - and finally puts it on D_1.

Your task is to write a program which simulates the behaviors of the students and the librarian, and evaluates the total cost of the overall process. Costs are associated with *accessing* a desk or the shelf, that is, putting/taking a book on/from it in the description above. The cost of an access is i for desk D_i and $m + 1$ for the shelf. That is, an access to $D_1, \cdots, D_m$, and the shelf costs $1, \cdots, m$, and $m + 1$, respectively. Costs of other actions are ignored.

Initially, no books are put on desks. No new students appear after opening the library.

Input

The input consists of multiple datasets. The end of the input is indicated by a line containing three zeros separated by a space. It is not a dataset.

The format of each dataset is as follows.

$$\begin{array}{l} m \quad c \quad n \\ k_1 \\ b_{11} \quad \ldots \quad b_{1k_1} \\ \vdots \\ k_n \\ b_{n1} \quad \ldots \quad b_{nk_n} \end{array}$$

Here, all data items are positive integers. m is the number of desks not exceeding 10. c is the number of books allowed to put on a desk, which does not exceed 30. n is the number of students not exceeding 100. k_i is the number of books requested by the i-th student, which does not exceed 50. b_{ij} is the ID number of the book requested by the i-th student on the j-th turn. No two books have the same ID number. Note that a student may request the same book more than once. b_{ij} is less than 100.

Here we show you an example of cost calculation for the following dataset.

```
3 1 2
```

```
3
60 61 62
2
70 60
```

In this dataset, there are 3 desks (D_1, D_2, D_3). At most 1 book can be put on each desk. The number of students is 2. The first student requests 3 books of which IDs are 60, 61, and 62, respectively, and the second student 2 books of which IDs are 70 and 60, respectively.

The calculation of the cost for this dataset is done as follows. First, for the first request of the first student, the librarian takes the book 60 from the shelf and puts it on D_1 and the first student goes to the end of the queue, costing 5. Next, for the first request of the second student, the librarian takes the book 70 from the shelf, puts it on D_2, moves the book 60 from D_1 to D_3, and finally moves the book 70 from D_2 to D_1, costing 13. Similarly, the cost for the books 61, 60, and 62, are calculated as 14, 12, 14, respectively. Therefore, the total cost is 58.

Output

For each dataset, output the total cost of processing all the requests, in a separate line.

Sample Input

```
2 1 1
1
50
2 1 2
1
50
1
60
2 1 2
2
60 61
1
70
4 2 3
3
60 61 62
1
70
2
80 81
3 1 2
3
60 61 62
2
```

```
70 60
1 2 5
2
87 95
3
96 71 35
2
68 2
3
3 18 93
2
57 2
2 2 1
5
1 2 1 3 1
0 0 0
```

Output for the Sample Input

```
4
16
28
68
58
98
23
```

Problem C
Colored Cubes
Input: C.txt

There are several colored cubes. All of them are of the same size but they may be colored differently. Each face of these cubes has a single color. Colors of distinct faces of a cube may or may not be the same.

Two cubes are said to be *identically colored* if some suitable rotations of one of the cubes give identical looks to both of the cubes. For example, two cubes shown in Figure 2 are identically colored. A set of cubes is said to be identically colored if every pair of them are identically colored.

A cube and its mirror image are not necessarily identically colored. For example, two cubes shown in Figure 3 are not identically colored.

You can make a given set of cubes identically colored by repainting some of the faces, whatever colors the faces may have. In Figure 4, repainting four faces makes the three cubes identically colored and repainting fewer faces will never do.

Your task is to write a program to calculate the minimum number of faces that needs to be repainted for a given set of cubes to become identically colored.

Input

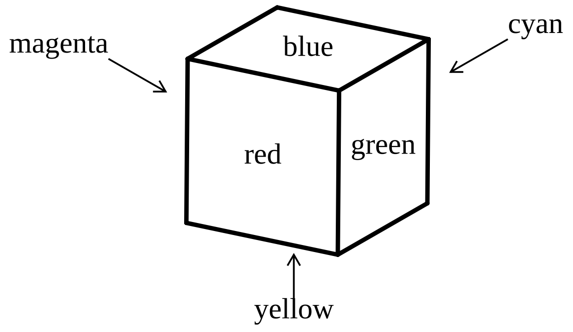
The input is a sequence of datasets. A dataset consists of a header and a body appearing in this order. A header is a line containing one positive integer n and the body following it consists of n lines. You can assume that $1 \leq n \leq 4$. Each line in a body contains six color names separated by a space. A color name consists of a word or words connected with a hyphen (-). A word consists of one or more lowercase letters. You can assume that a color name is at most 24-characters long including hyphens.

A dataset corresponds to a set of colored cubes. The integer n corresponds to the number of cubes. Each line of the body corresponds to a cube and describes the colors of its faces. Color names in a line is ordered in accordance with the numbering of faces shown in Figure 5. A line

$color_1$ $color_2$ $color_3$ $color_4$ $color_5$ $color_6$

corresponds to a cube colored as shown in Figure 6.

The end of the input is indicated by a line containing a single zero. It is not a dataset nor a part of a dataset.



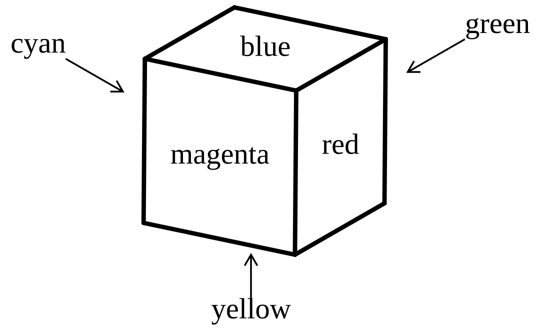
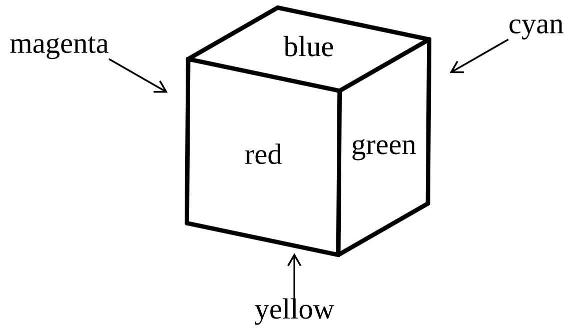


Figure 2: Identically colored cubes



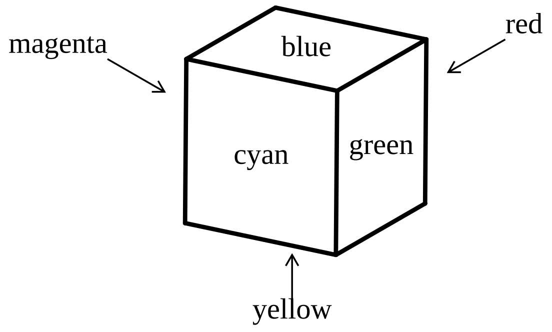


Figure 3: cubes that are not identically colored

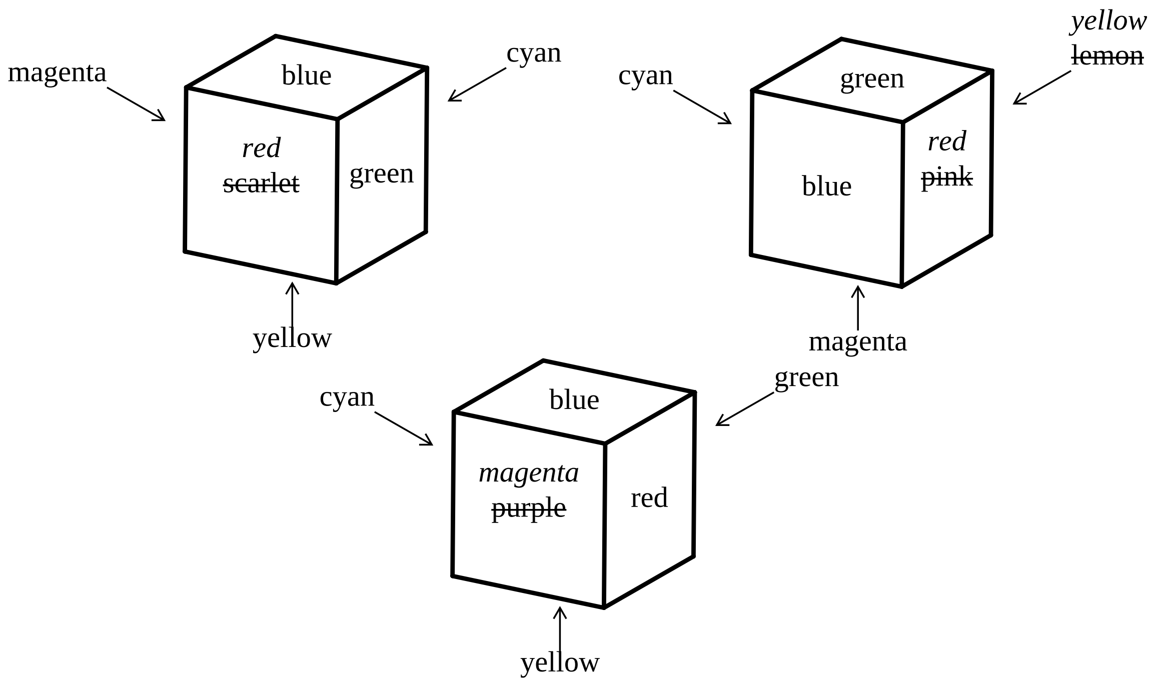


Figure 4: An example of recoloring

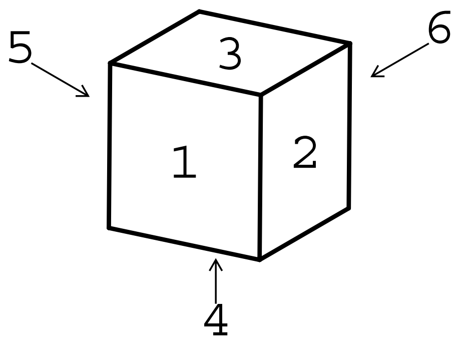


Figure 5: Numbering of faces

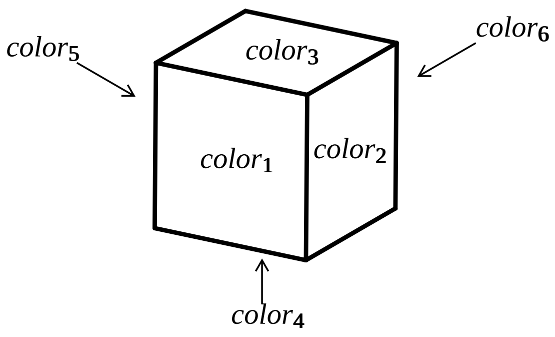


Figure 6: Coloring

Output

For each dataset, output a line containing the minimum number of faces that need to be repainted to make the set of cubes identically colored.

Sample Input

```
3
scarlet green blue yellow magenta cyan
blue pink green magenta cyan lemon
purple red blue yellow cyan green
2
red green blue yellow magenta cyan
cyan green blue yellow magenta red
2
red green gray gray magenta cyan
cyan green gray gray magenta red
2
red green blue yellow magenta cyan
magenta red blue yellow cyan green
3
red green blue yellow magenta cyan
cyan green blue yellow magenta red
magenta red blue yellow cyan green
3
blue green green green green blue
green blue blue green green green
green green green green green sea-green
3
red yellow red yellow red yellow
red red yellow yellow red yellow
red red red red red red
4
violet violet salmon salmon salmon salmon
violet salmon salmon salmon salmon violet
violet violet salmon salmon violet violet
violet violet violet violet salmon salmon
1
red green blue yellow magenta cyan
4
magenta pink red scarlet vermilion wine-red
aquamarine blue cyan indigo sky-blue turquoise-blue
blond cream chrome-yellow lemon olive yellow
chrome-green emerald-green green olive vilidian sky-blue
0
```

Output for the Sample Input

```
4
2
0
0
2
3
4
4
0
16
```

Problem D
Organize Your Train
Input: D.txt

In the good old Hachioji railroad station located in the west of Tokyo, there are several parking lines, and lots of freight trains come and go every day.

All freight trains travel at night, so these trains containing various types of cars are settled in your parking lines early in the morning. Then, during the daytime, you must reorganize cars in these trains according to the request of the railroad clients, so that every line contains the "right" train, i.e. the right number of cars of the right types, in the right order.

As shown in Figure 7, all parking lines run in the East-West direction. There are exchange lines connecting them through which you can move cars. An exchange line connects two ends of different parking lines. Note that an end of a parking line can be connected to many ends of other lines. Also note that an exchange line may connect the East-end of a parking line and the West-end of another.

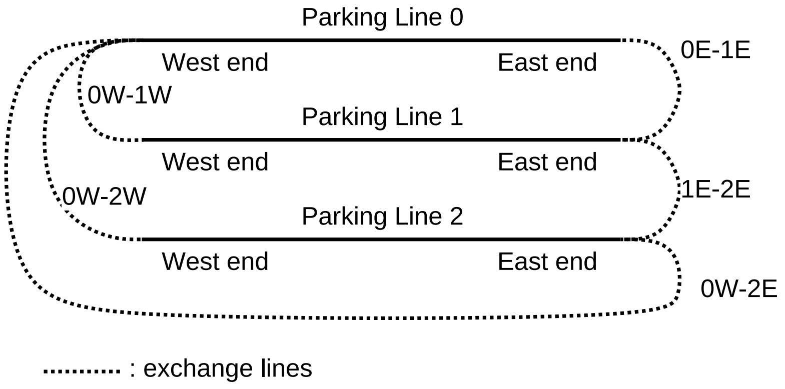


Figure 7: Parking lines and exchange lines

Cars of the same type are not discriminated between each other. The cars are symmetric, so directions of cars don't matter either.

You can divide a train at an arbitrary position to make two sub-trains and move one of them through an exchange line connected to the end of its side. Alternatively, you may move a whole train as is without dividing it. Anyway, when a (sub-) train arrives at the destination parking line and the line already has another train in it, they are coupled to form a longer train.

Your superautomatic train organization system can do these without any help of locomotive engines. Due to the limitation of the system, trains cannot stay on exchange lines; when you start moving a (sub-) train, it must arrive at the destination parking line before moving another train.

In what follows, a letter represents a car type and a train is expressed as a sequence of letters. For example in Figure 8, from an initial state having a train "`aabbccdee`" on line 0 and no trains on other lines, you can make "`bbaadeecc`" on line 2 with the four moves shown in the figure.

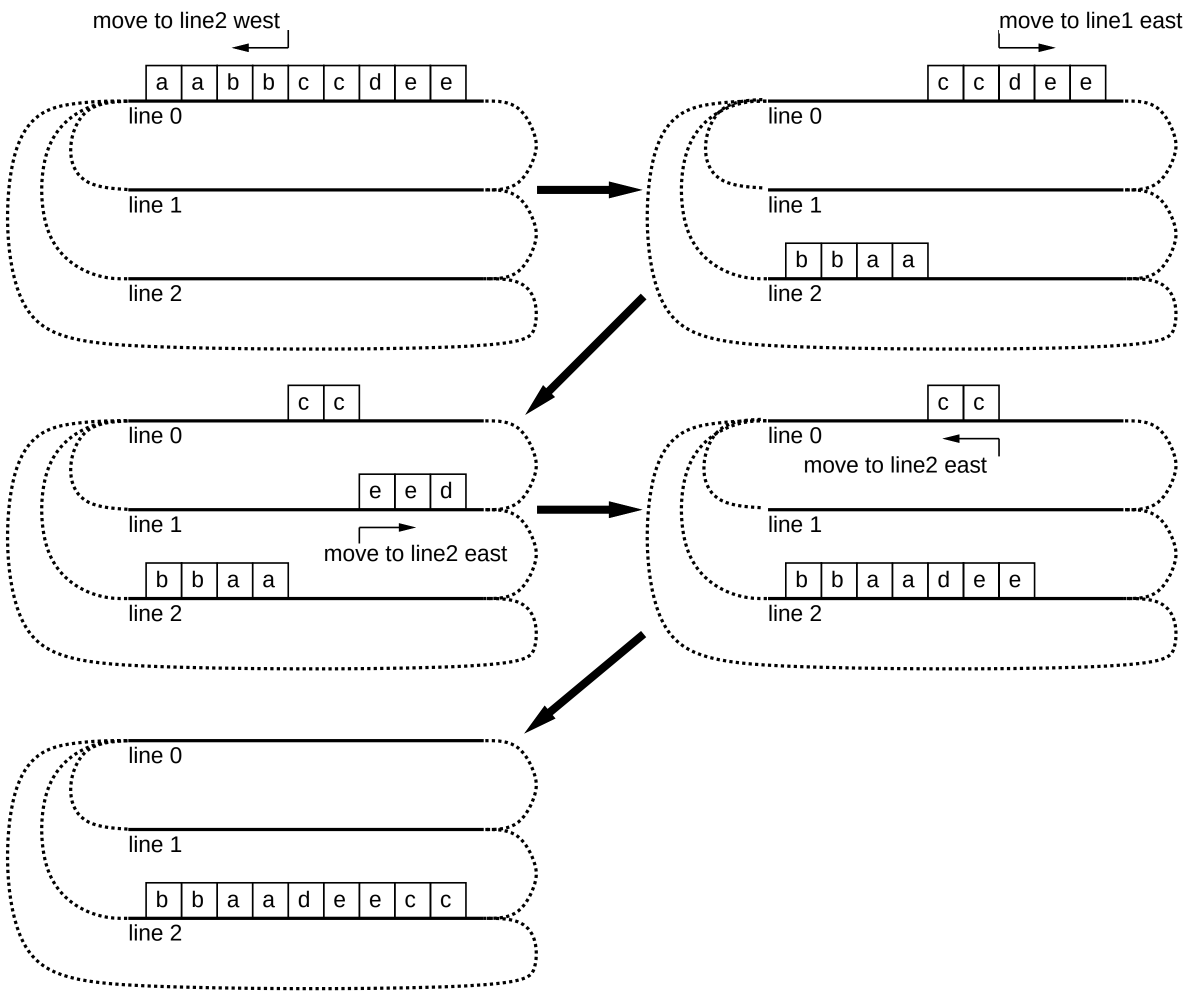


Figure 8: An example movement sequence

To cut the cost out, your boss wants to minimize the number of (sub-) train movements. For example, in the case of Figure 8, the number of movements is 4 and this is the minimum.

Given the configurations of the train cars in the morning (arrival state) and evening (departure state), your job is to write a program to find the optimal train reconfiguration plan.

Input

The input consists of one or more datasets. A dataset has the following format:

$x\ y$
$p_1P_1\ q_1Q_1$
$p_2P_2\ q_2Q_2$
$\vdots$
$p_yP_y\ q_yQ_y$
s_0
s_1
$\vdots$
s_{x-1}
t_0
t_1
$\vdots$
t_{x-1}

x is the number of parking lines, which are numbered from 0 to $x-1$. y is the number of exchange lines. Then y lines of the exchange line data follow, each describing two ends connected by the exchange line; p_i and q_i are integers between 0 and $x-1$ which indicate parking line numbers, and P_i and Q_i are either "`E`" (East) or "`W`" (West) which indicate the ends of the parking lines.

Then x lines of the arrival (initial) configuration data, $s_0, \cdots, s_{x-1}$, and x lines of the departure (target) configuration data, $t_0, \cdots t_{x-1}$, follow. Each of these lines contains one or more lowercase letters "`a`", "`b`", $\cdots$, "`z`", which indicate types of cars of the train in the corresponding parking line, in west to east order, or alternatively, a single "`-`" when the parking line is empty.

You may assume that x does not exceed 4, the total number of cars contained in all the trains does not exceed 10, and every parking line has sufficient length to park all the cars.

You may also assume that each dataset has at least one solution and that the minimum number of moves is between one and six, inclusive.

Two zeros in a line indicate the end of the input.

Output

For each dataset, output the number of moves for an optimal reconfiguration plan, in a separate line.

Sample Input

```
3 5
0W 1W
0W 2W
0W 2E
0E 1E
1E 2E
aabbccdee
-
-
-
-
bbaadeecc
3 3
0E 1W
1E 2W
2E 0W
aabb
bbcc
aa
bbbb
cc
aaaa
3 4
0E 1W
0E 2E
1E 2W
2E 0W
ababab
-
-
aaabbb
-
-
0 0
```

Output for the Sample Input

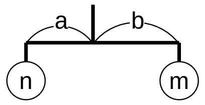
```
4
2
5
```

Problem E
Mobile Computing
Input: E.txt

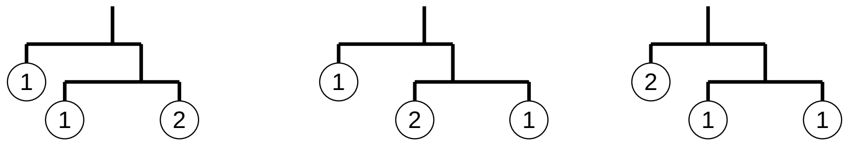
There is a mysterious planet called Yaen, whose space is 2-dimensional. There are many beautiful stones on the planet, and the Yaen people love to collect them. They bring the stones back home and make nice mobile arts of them to decorate their 2-dimensional living rooms.

In their 2-dimensional world, a mobile is defined recursively as follows:

- a stone hung by a string, or
- a rod of length 1 with two sub-mobiles at both ends; the rod is hung by a string at the center of gravity of sub-mobiles. When the weights of the sub-mobiles are n and m, and their distances from the center of gravity are a and b respectively, the equation $n \times a = m \times b$ holds.



For example, if you got three stones with weights 1, 1, and 2, here are some possible mobiles and their widths:



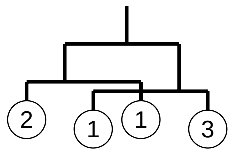
width = 1 + (1/3) width = 1 + (2/3) width = 1 + (1/2)

Given the weights of stones and the width of the room, your task is to design the widest possible mobile satisfying both of the following conditions.

- It uses all the stones.
- Its width is less than the width of the room.

You should ignore the widths of stones.

In some cases two sub-mobiles hung from both ends of a rod might overlap (see the figure on the right). Such mobiles are acceptable. The width of the example is (1/3) + 1 + (1/4).



Input

The first line of the input gives the number of datasets. Then the specified number of datasets follow. A dataset has the following format.

$$\begin{array}{l} r \\ s \\ w_1 \\ \vdots \\ w_s \end{array}$$

r is a decimal fraction representing the width of the room, which satisfies $0 < r < 10$. s is the number of the stones. You may assume $1 \leq s \leq 6$. w_i is the weight of the i-th stone, which is an integer. You may assume $1 \leq w_i \leq 1000$.

You can assume that no mobiles whose widths are between $r - 0.00001$ and $r + 0.00001$ can be made of given stones.

Output

For each dataset in the input, one line containing a decimal fraction should be output. The decimal fraction should give the width of the widest possible mobile as defined above. An output line should not contain extra characters such as spaces.

In case there is no mobile which satisfies the requirement, answer -1 instead.

The answer should not have an error greater than 0.00000001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

Sample Input

```
5
1.3
3
1
2
1
1.4
3
1
2
1
2.0
3
1
```

```
2
1
1.59
4
2
1
1
3
1.7143
4
1
2
3
5
```

Output for the Sample Input

```
-1
1.3333333333333335
1.6666666666666667
1.5833333333333335
1.7142857142857142
```

Problem F
Atomic Car Race
Input: F.txt

In the year 2020, a race of atomically energized cars will be held. Unlike today's car races, fueling is not a concern of racing teams. Cars can run throughout the course without any refueling. Instead, the critical factor is tire (tyre). Teams should carefully plan where to change tires of their cars.

The race is a road race having n checkpoints in the course. Their distances from the start are a_1, a_2, $\cdots$, and a_n (in kilometers). The n-th checkpoint is the goal. At the i-th checkpoint ($i < n$), tires of a car can be changed. Of course, a team can choose whether to change or not to change tires at each checkpoint. It takes b seconds to change tires (including overhead for braking and accelerating). There is no time loss at a checkpoint if a team chooses not to change tires.

A car cannot run fast for a while after a tire change, because the temperature of tires is lower than the designed optimum. After running long without any tire changes, on the other hand, a car cannot run fast because worn tires cannot grip the road surface well. The time to run an interval of one kilometer from x to $x + 1$ is given by the following expression (in seconds). Here x is a nonnegative integer denoting the distance (in kilometers) from the latest checkpoint where tires are changed (or the start). r, v, e and f are given constants.

$$\begin{array}{ll} 1/(v - e \times (x - r)) & \text{(if } x \geq r\text{)} \\ 1/(v - f \times (r - x)) & \text{(if } x < r\text{)} \end{array}$$

Your mission is to write a program to determine the best strategy of tire changes which minimizes the total time to the goal.

Input

The input consists of multiple datasets each corresponding to a race situation. The format of a dataset is as follows.

n
$a_1\ a_2\ \ldots\ a_n$
b
$r\ v\ e\ f$

The meaning of each of the input items is given in the problem statement. If an input line contains two or more input items, they are separated by a space.

n is a positive integer not exceeding 100. Each of a_1, a_2, $\cdots$, and a_n is a positive integer satisfying $0 < a_1 < a_2 < \ldots < a_n \leq 10000$. b is a positive decimal fraction not exceeding 100.0. r is a nonnegative integer satisfying $0 \leq r \leq a_n - 1$. Each of v, e and f is a positive decimal fraction. You can assume that $v - e \times (a_n - 1 - r) \geq 0.01$ and $v - f \times r \geq 0.01$.

The end of the input is indicated by a line with a single zero.

Output

For each dataset in the input, one line containing a decimal fraction should be output. The decimal fraction should give the elapsed time at the goal (in seconds) when the best strategy is taken. An output line should not contain extra characters such as spaces.

The answer should not have an error greater than 0.001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

Sample Input

```
2
2 3
1.0
1 1.0 0.1 0.3
5
5 10 15 20 25
0.15
1 1.0 0.04 0.5
10
1783 3640 3991 4623 5465 5481 6369 6533 6865 8425
4.172
72 59.4705 0.0052834 0.0611224
0
```

Output for the Sample Input

```
3.5397
31.9249
168.6682
```

Problem G
Network Mess
Input: G.txt

Gilbert is the network admin of Ginkgo company. His boss is mad about the messy network cables on the floor. He finally walked up to Gilbert and asked the lazy network admin to illustrate how computers and switches are connected. Since he is a programmer, he is very reluctant to move throughout the office and examine cables and switches with his eyes. He instead opted to get this job done by measurement and a little bit of mathematical thinking, sitting down in front of his computer all the time. Your job is to help him by writing a program to reconstruct the network topology from measurements.

There are a known number of computers and an unknown number of switches. Each computer is connected to one of the switches via a cable and to nothing else. Specifically, a computer is never connected to another computer directly, or never connected to two or more switches. Switches are connected via cables to form a tree (a connected undirected graph with no cycles). No switches are 'useless.' In other words, each switch is on the path between at least one pair of computers.

All in all, computers and switches together form a tree whose leaves are computers and whose internal nodes switches (See Figure 9).

Gilbert measures the distances between *all pairs of computers*. The distance between two computers is simply the number of switches on the path between the two, plus one. Or equivalently, it is the number of cables used to connect them. You may wonder how Gilbert can actually obtain these distances solely based on measurement. Well, he can do so by a very sophisticated statistical processing technique he invented. Please do not ask the details.

You are therefore given a matrix describing distances between leaves of a tree. Your job is to construct the tree from it.

Input

The input is a series of distance matrices, followed by a line consisting of a single '0'. Each distance matrix is formatted as follows.

$$\begin{array}{cccc} N & & & \\ a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{array}$$

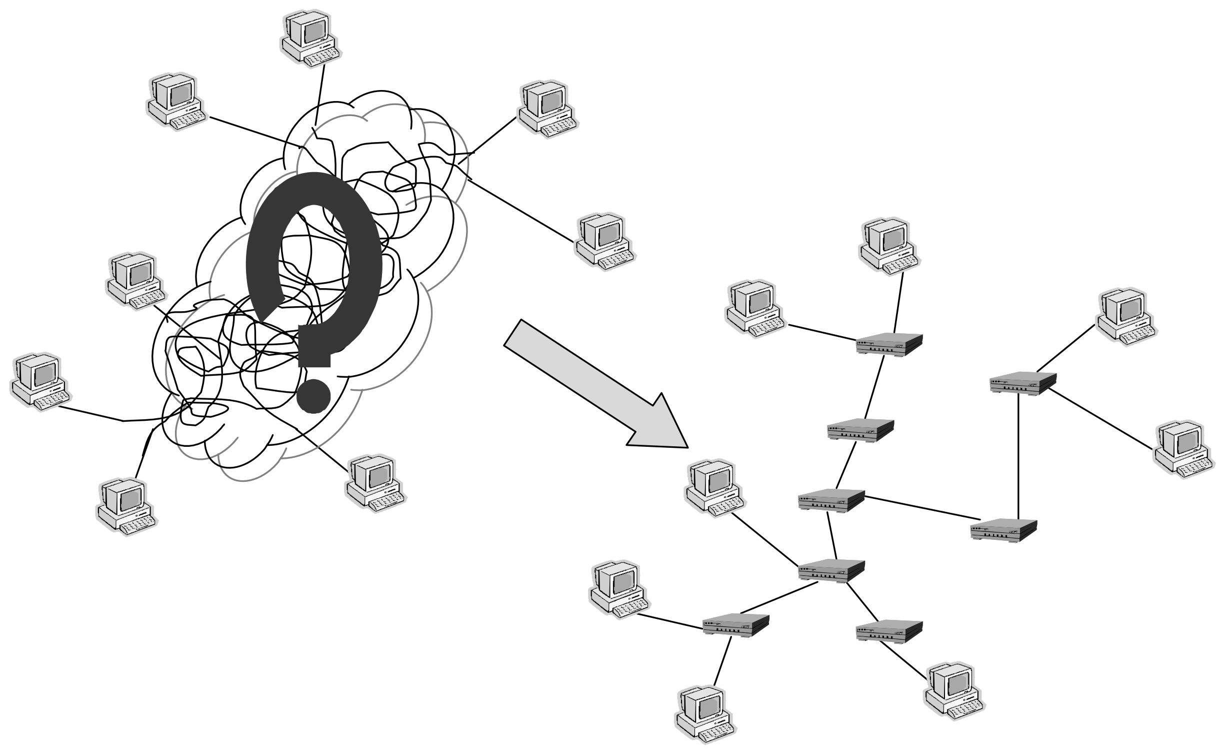


Figure 9: Computers and Switches

N is the size, i.e. the number of rows and the number of columns, of the matrix. a_{ij} gives the distance between the i-th leaf node (computer) and the j-th. You may assume $2 \leq N \leq 50$ and the matrix is symmetric whose diagonal elements are all zeros. That is, $a_{ii} = 0$ and $a_{ij} = a_{ji}$ for each i and j. Each non-diagonal element a_{ij} ($i \neq j$) satisfies $2 \leq a_{ij} \leq 30$. You may assume there is always a solution. That is, there is a tree having the given distances between leaf nodes.

Output

For each distance matrix, find a tree having the given distances between leaf nodes. Then output the degree of each internal node (i.e. the number of cables adjoining each switch), all in a single line and in ascending order. Numbers in a line should be separated by a single space. A line should not contain any other characters, including trailing spaces.

Sample Input

```
4
  0  2  2  2
  2  0  2  2
  2  2  0  2
  2  2  2  0
4
  0  2  4  4
  2  0  4  4
  4  4  0  2
  4  4  2  0
2
  0 12
 12  0
0
```

Output for the Sample Input

```
4
2 3 3
2 2 2 2 2 2 2 2 2 2 2
```

Problem H
Bingo
Input: H.txt

A Bingo game is played by one gamemaster and several players. At the beginning of a game, each player is given a card with $M \times M$ numbers in a matrix (See Figure 10).

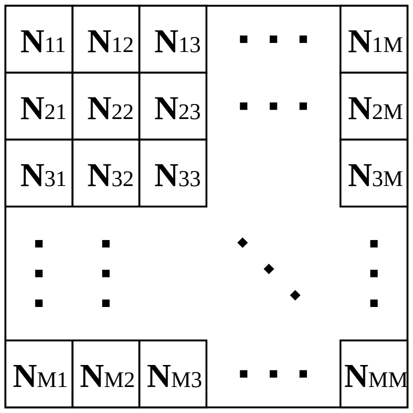


Figure 10: A Card

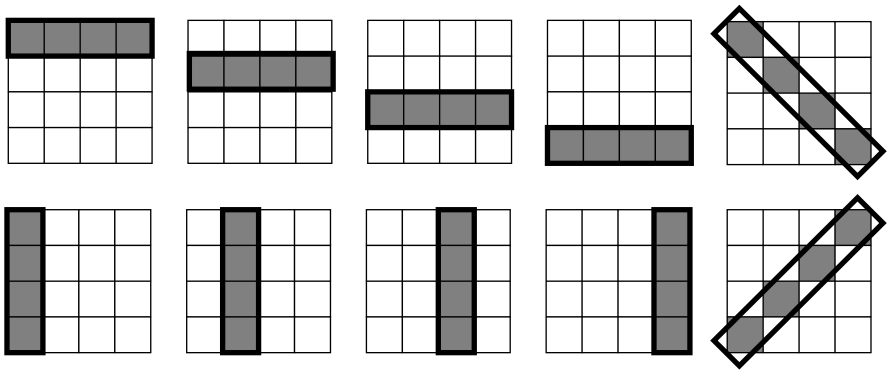


Figure 11: Bingo patterns of 4×4 card

As the game proceeds, the gamemaster announces a series of numbers one by one. Each player punches a hole in his card on the announced number, if any.

When at least one 'Bingo' is made on the card, the player wins and leaves the game. The 'Bingo' means that all the M numbers in a line are punched vertically, horizontally or diagonally (See Figure 11).

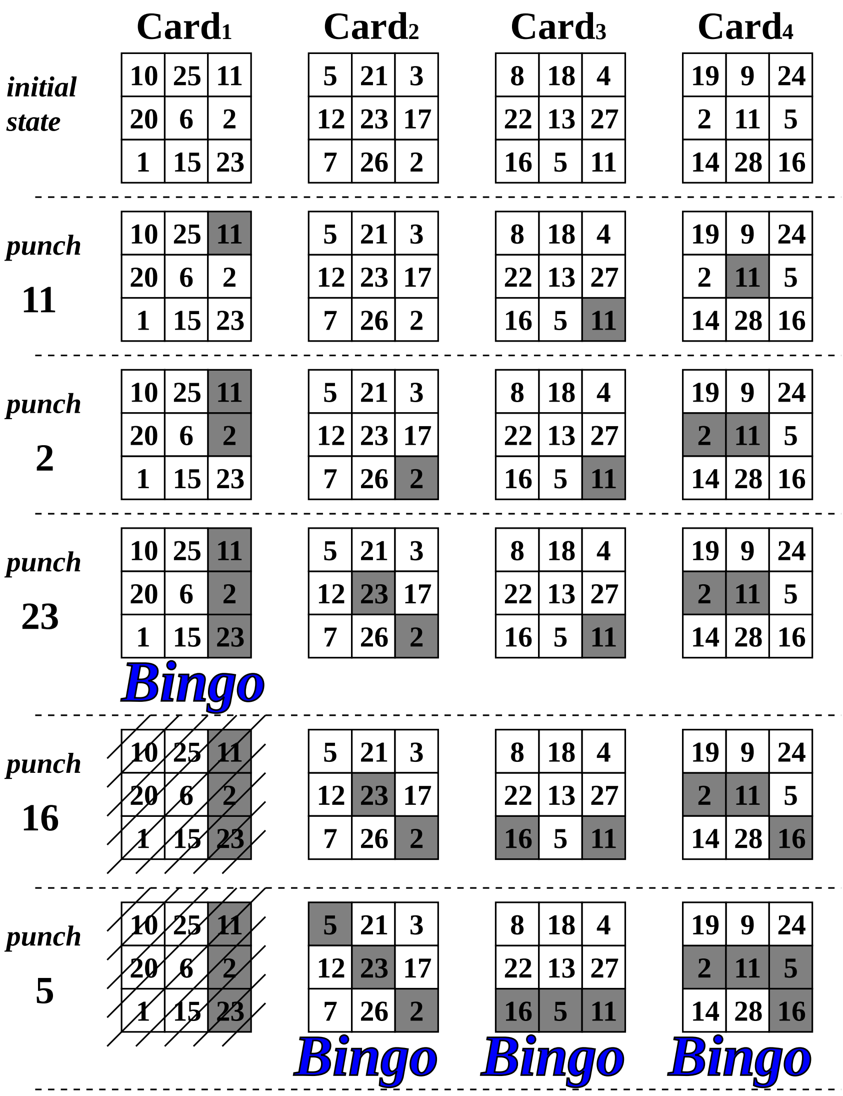


Figure 12: Example of Bingo Game Process

The gamemaster continues announcing numbers until all the players make a Bingo.

In the ordinary Bingo games, the gamemaster chooses numbers by a random process and has no control on them. But in this problem the gamemaster knows all the cards at the beginning of the game and controls the game by choosing the number sequence to be announced at his will.

Specifically, he controls the game to satisfy the following condition.

$$\text{Card}_i \text{ makes a Bingo no later than Card}_j\text{, for } i < j. \qquad (*)$$

Figure 12 shows an example of how a game proceeds. The gamemaster cannot announce '5' before '16', because Card$_4$ makes a Bingo before Card$_2$ and Card$_3$, violating the condition $(*)$.

Your job is to write a program which finds the minimum length of such sequence of numbers for the given cards.

Input

The input consists of multiple datasets. The format of each dataset is as follows.

$$\begin{array}{cccccccccccc}
P & M & & & & & & & & & & \\
N^1_{11} & N^1_{12} & \dots & N^1_{1M} & N^1_{21} & N^1_{22} & \dots & N^1_{2M} & \dots & N^1_{M1} & N^1_{M2} & \dots & N^1_{MM} \\
N^2_{11} & N^2_{12} & \dots & N^2_{1M} & N^2_{21} & N^2_{22} & \dots & N^2_{2M} & \dots & N^2_{M1} & N^2_{M2} & \dots & N^2_{MM} \\
 & & & & & & \vdots & & & & & & \\
N^P_{11} & N^P_{12} & \dots & N^P_{1M} & N^P_{21} & N^P_{22} & \dots & N^P_{2M} & \dots & N^P_{M1} & N^P_{M2} & \dots & N^P_{MM}
\end{array}$$

All data items are integers. P is the number of the cards, namely the number of the players. M is the number of rows and the number of columns of the matrix on each card. N^k_{ij} means the number written at the position (i, j) on the k-th card. If $(i, j) \neq (p, q)$, then $N^k_{ij} \neq N^k_{pq}$. The parameters P, M, and N satisfy the conditions $2 \leq P \leq 4$, $3 \leq M \leq 4$, and $0 \leq N^k_{ij} \leq 99$.

The end of the input is indicated by a line containing two zeros separated by a space. It is not a dataset.

Output

For each dataset, output the minimum length of the sequence of numbers which satisfy the condition $(*)$. Output a zero if there are no such sequences. Output for each dataset must be printed on a separate line.

Sample Input

```
4 3
10 25 11 20 6 2 1 15 23
```

```
5 21 3 12 23 17 7 26 2
8 18 4 22 13 27 16 5 11
19 9 24 2 11 5 14 28 16
4 3
12 13 20 24 28 32 15 16 17
12 13 21 25 29 33 16 17 18
12 13 22 26 30 34 17 18 15
12 13 23 27 31 35 18 15 16
4 3
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
4 4
2 6 9 21 15 23 17 31 33 12 25 4 8 24 13 36
22 18 27 26 35 28 3 7 11 20 38 16 5 32 14 29
26 7 16 29 27 3 38 14 18 28 20 32 22 35 11 5
36 13 24 8 4 25 12 33 31 17 23 15 21 9 6 2
0 0
```

Output for the Sample Input

```
5
4
12
0
```

For your convenience, sequences satisfying the condition (∗) for the first three datasets are shown below. There may be other sequences of the same length satisfying the condition, but no shorter.

```
11, 2, 23, 16, 5
15, 16, 17, 18
11, 12, 13, 21, 22, 23, 31, 32, 33, 41, 42, 43
```

Problem I
Shy Polygons
Input: I.txt

You are given two solid polygons and their positions on the *xy*-plane. You can move one of the two along the *x*-axis (they can overlap during the move). You cannot move it in other directions. The goal is to place them as compactly as possible, subject to the following condition: the distance between any point in one polygon and any point in the other must not be smaller than a given minimum distance L.

We define the *width* of a placement as the difference between the maximum and the minimum x-coordinates of all points in the two polygons.

Your job is to write a program to calculate the minimum width of placements satisfying the above condition.

Let's see an example. If the polygons in Figure 13 are placed with $L = 10.0$, the result will be 100. Figure 14 shows one of the optimal placements.

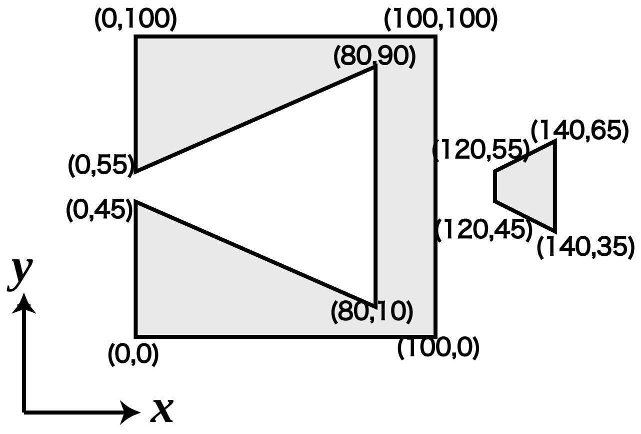


Figure 13: Initial position of the two polygons

Input

The input consists of multiple datasets. Each dataset is given in the following format.

> L
> $Polygon_1$
> $Polygon_2$

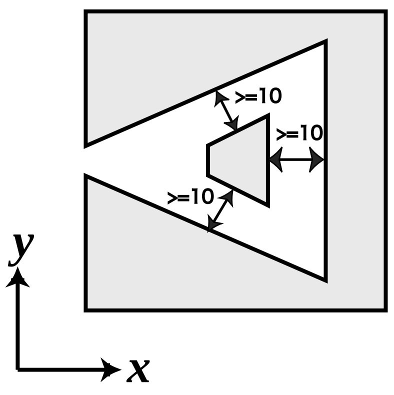


Figure 14: One of the optimal placements ($L = 10.0$)

L is a decimal fraction, which means the required distance of two polygons. L is greater than 0.1 and less than 50.0.

The format of each polygon is as follows.

n
x_1 y_1
x_2 y_2
$\vdots$
x_n y_n

n is a positive integer, which represents the number of vertices of the polygon. n is greater than 2 and less than 15.

Remaining lines represent the vertices of the polygon. A vertex data line has a pair of nonnegative integers which represent the x- and y-coordinates of a vertex. x- and y-coordinates are separated by a single space, and y-coordinate is immediately followed by a newline. x and y are less than 500.

Edges of the polygon connect vertices given in two adjacent vertex data lines, and vertices given in the last and the first vertex data lines. You may assume that the vertices are given in the counterclockwise order, and the contours of polygons are simple, i.e. they do not cross nor touch themselves.

Also, you may assume that the result is not sensitive to errors. In concrete terms, for a given pair of polygons, the minimum width is a function of the given minimum distance l. Let us denote the function $w(l)$. Then you can assume that $|w(L \pm 10^{-7}) - w(L)| < 10^{-4}$.

The end of the input is indicated by a line that only contains a zero. It is not a part of a dataset.

Output

The output should consist of a series of lines each containing a single decimal fraction. Each number should indicate the minimum width for the corresponding dataset. The answer should not have an error greater than 0.0001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

Sample Input

```
10.5235
3
0 0
100 100
0 100
4
0 50
20 50
20 80
0 80
10.0
4
120 45
140 35
140 65
120 55
8
0 0
100 0
100 100
0 100
0 55
80 90
80 10
0 45
10.0
3
0 0
1 0
0 1
3
0 100
1 101
0 101
10.0
3
0 0
```

```
1 0
0 100
3
0 50
100 50
0 51
0
```

Output for the Sample Input

```
114.882476
100
1
110.5005
```

Problem J. Snapper Chain

The *Snapper* is a clever little device that, on one side, plugs its input plug into an output socket, and, on the other side, exposes an output socket for plugging in a light or other device.

When a *Snapper* is in the ON state and is receiving power from its input plug, then the device connected to its output socket is receiving power as well. When you snap your fingers — making a clicking sound — any *Snapper* receiving power at the time of the snap toggles between the ON and OFF states.

In hopes of destroying the universe by means of a singularity, I have purchased n *Snapper* devices and chained them together by plugging the first one into a power socket, the second one into the first one, and so on. The light is plugged into the n-th *Snapper*.

Initially, all the *Snappers* are in the OFF state, so only the first one is receiving power from the socket, and the light is off. I snap my fingers once, which toggles the first *Snapper* into the ON state and gives power to the second one. I snap my fingers again, which toggles both *Snappers* and then promptly cuts power off from the second one, leaving it in the ON state, but with no power. I snap my fingers the third time, which toggles the first *Snapper* again and gives power to the second one. Now both *Snappers* are in the ON state, and if my light is plugged into the second *Snapper* it will be on.

I keep doing this for hours. Will the light be on or off after I have snapped my fingers k times? The light is on if and only if it's receiving power from the *Snapper* it's plugged into.

Input

The first line of the input gives a single integer t ($1 \le t \le 10000$) — the number of test cases.

Each of the following t lines contains two integers n and k ($1 \le n \le 30$, $0 \le k \le 10^{18}$).

Output

For each test case, output one line containing "ON" or "OFF", indicating the state of the light bulb.

Examples

standard input	standard output

4
1 0
1 1
4 0
4 47
	OFF
	ON
	OFF
	ON

Problem K1. Fair Warning

On our planet, Jamcode IX, three Great Events occurred. They happened 26000, 11000 and 6000 slarboseconds ago. In 4000 slarboseconds, the amount of time since all of those events will be multiples of 5000 slarboseconds, the largest possible amount... and the apocalypse will come.

Luckily for you, you live on Jamcode X! The apocalypse came on Jamcode IX less than a year ago. But Jamcode X has a worrying prophecy: "After the moment of reckoning, on the first *optimum anniversary* of the n Great Events, the apocalypse will come. 64 bits will not save you. You have been warned."

The people of Jamcode X are very concerned by this prophecy. All of the Great Events have already happened, and their times have been measured to the nearest slarbosecond; but nobody knows when their *optimum anniversary* will occur. After studying the diary of a scientist from Jamcode IX, scientists working on the problem have come up with a theory:

The *moment of reckoning* is now, the moment you solve this problem. At some time $y \ge 0$ slarboseconds from now, the number of slarboseconds since each of the Great Events will be divisible by some maximum number t. If you can find the smallest value of y that gives this largest possible t, that will give you the *optimum anniversary* when the apocalypse will come.

On Jamcode IX, for example, there were 3 Great Events and they happened 26000, 11000 and 6000 slarboseconds before the moment of reckoning. 4000 slarboseconds later, the amount of time since each event was a multiple of $t = 5000$ slarboseconds, and the apocalypse came.

Your job is to compute the amount of time until the apocalypse comes.

Input

The first line of the input gives a single integer n ($2 \le n \le 3$).

The second line of the input gives n space-separated integers t_i ($1 \le t_i \le 10^8$), the number of slarboseconds since Great Event i occurred. It is guaranteed that $t_i \ne t_j$ for some i, j.

Output

Output one line containing the minimum number of slarboseconds until $t_i + y$ is a multiple of the largest possible integer factor t for all i.

Examples

standard input	standard output

3
26000000 11000000
6000000
	4000000

3 1 10 11	0
2 8000001 9000001	999999

Notes

Epilogue

Fortunately for the peoples of the Jamcode system, "the apocalypse" turned out to be a mistranslation of "the giant party." Nobody from Jamcode IX bothered to pass this along, because they were having so much fun.

Problem K2. Fair Warning

The statement of this problem is the same with the statement of the problem **K1**, but the constraints are larger: $2 \le N \le 1000$, $1 \le t_i \le 10^{50}$.

Problem L1. Theme Park

Roller coasters are so much fun! It seems like everybody who visits the theme park wants to ride the roller coaster. Some people go alone; other people go in groups, and don't want to board the roller coaster unless they can all go together. And *everyone* who rides the roller coaster wants to ride again. A ride costs 1 Euro per person; your job is to figure out how much money the roller coaster will make today.

The roller coaster can hold k people at once. People queue for it in groups. Groups board the roller coaster, one at a time, until there are no more groups left or there is no room for the next group; then the roller coaster goes, whether it's full or not. Once the ride is over, all of its passengers re-queue in the same order. The roller coaster will run r times in a day.

For example, suppose $r = 4$, $k = 6$, and there are four groups of people with sizes: 1, 4, 2, 1. The first time the roller coaster goes, the first two groups $[1, 4]$ will ride, leaving an empty seat (the group of 2 won't fit, and the group of 1 can't go ahead of them). Then they'll go to the back of the queue, which now looks like 2, 1, 1, 4. The second time, the coaster will hold 4 people: $[2, 1, 1]$. Now the queue looks like 4, 2, 1, 1. The third time, it will hold 6 people: $[4, 2]$. Now the queue looks like $[1, 1, 4, 2]$. Finally, it will hold 6 people: $[1, 1, 4]$. The roller coaster has made a total of 21 Euros!

Input

The first line of the input contains three space-separated integers r, k and n ($1 \le r \le 1000$, $1 \le k \le 100$, $1 \le n \le 10$).

The second line contains n space-separated integers g_i ($1 \le g_i \le 10$), each of which is the size of a group that wants to ride. g_0 is the size of the first group, g_1 is the size of the second group, etc. It is guaranteed that all $g_i \le k$.

Output

Output one line containing the number of Euros made by the roller coaster.

Examples

standard input	standard output
4 6 4 1 4 2 1	21
100 10 1 1	100
5 5 10 2 4 2 3 4 2 1 2 1 3	20

Problem L2. Theme Park

The statement of this problem is the same with the statement of the problem **L1**, but the constraints are larger: $1 \le r \le 10^8$, $1 \le k \le 10^9$, $1 \le n \le 1000$, $1 \le g_i \le 10^7$.