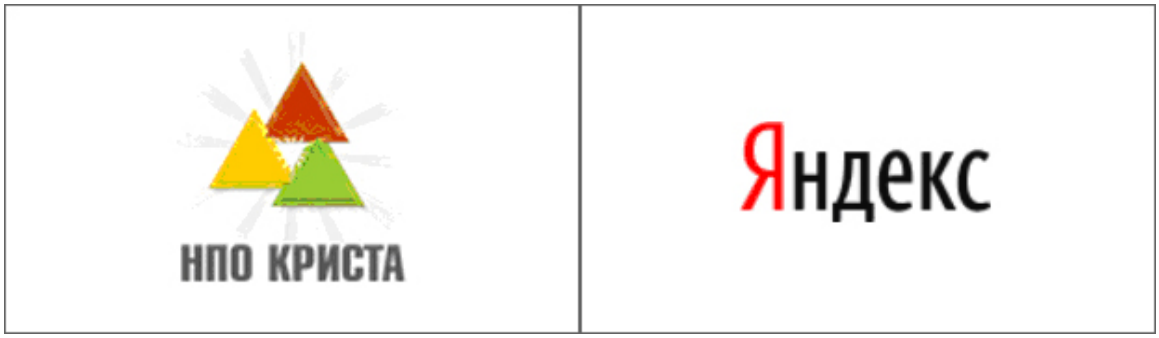
We can all benefit by doing occasional "toy" programs, when artificial restrictions are set up, so that we are forced to push our abilities to the limit. … The art of tackling miniproblems with all our energy will sharpen our talents for the real problems.
Donald E. Knuth

Quarterfinal

Central region of Russia

Rybinsk, October 17-19-2017

Input: standard input
Output: standard output
Memory: 256 Mb, **Time: 1 sec / test**



A. Normal magic square

A normal magic square of order n is an $n \times n$ grid (where n is the number of cells on each side), filled with distinct positive integers in the range from 1 to n^2, so that each cell contains a different integer and the sum of the integers in each row, column, and diagonal (primary and secondary) is equal.
Normal magic squares exist for all orders except 2×2 (order n=2). The 1×1 magic square, with only one cell containing the number 1, is called trivial. The minimal non-trivial case of order 3 is shown below.

2	7	6
9	5	1
4	3	8

You can practice building normal magic squares in your spare time, but now your task is to write a program that returns the sum of numbers in the first line, given the order of the square.

Input

The input file contains one positive integer n – the order of the square.

Output

In the output file, write down the sum of the elements on the first line of an $n \times n$ normal magic square.

Limitations

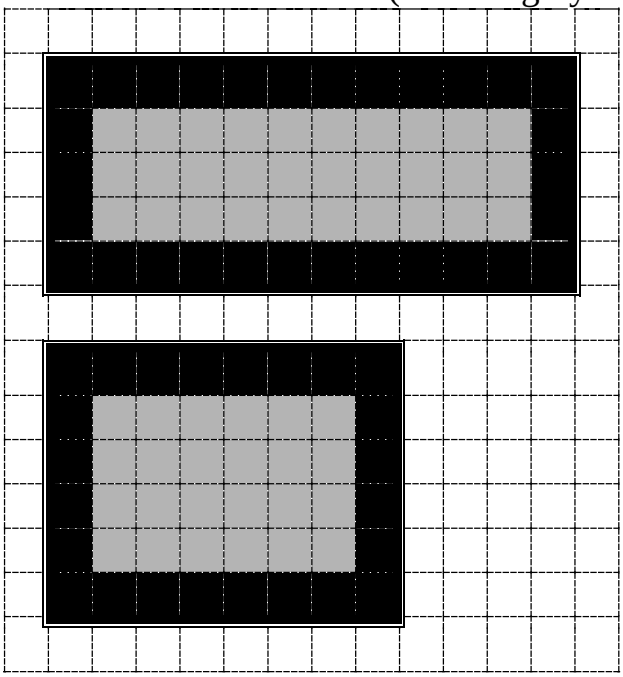
$1 \le n \le 1000$, $n \ne 2$, n – positive integer.

Examples

Input	Output
3	15
1	1

B. Rectangles

Imagine rectangles drawn on a piece of squared paper, their sides matching the lines of the grid. In such case, the cells within a rectangle are either external if they are adjacent to the sides (marked black in the figure below) or internal if they are not attached to the sides (marked grey in the figure).



Calculate how many rectangles there are with the number of external cells exactly ***n*** times smaller than the number of internal cells. Print the sizes of all appropriate rectangles in the ascending order based on the length of the smaller side.

Input

The input file contains a single positive integer $1 \le n \le 10^9$.

Output

In the first output line, print ***k*** – the number of rectangles meeting the above criterion. The next ***k*** lines must contain pairs of numbers representing the sides of appropriate rectangles (first the smaller side then the biggest one) in the ascending order based on the length of the smaller side.

Limitations

$1 \le n \le 10^9$.

Examples

Input	Output
1	2 5 12 6 8
2	4 7 30 8 18 9 14 10 12

C. Magic Football

Magic football is very popular in the Magic Land. In this game, the only way to hit the ball is using your magic skills. Unfortunately, players become exhausted after their magic efforts so they have to be substituted quite often. According to the rules of the game, each combination of players can play for exactly one minute. Each time only one player is substituted. It is prohibited to repeat the same team composition twice during the game.

The team from the Emerald City has ***n*** players. According to the rules of magic football, ***k*** players must be present in the field at any given moment. The game lasts ***t*** minutes.

Example: let ***n*** = 5, ***k*** = 3; in this case, 10 different team composition options are possible.

Minute No.	Team composition (numbers of players)	Substitution at the end of the minute
1	1 2 3	change 3 to 4
2	1 2 4	change 2 to 3
3	1 3 4	change 4 to 5
4	1 3 5	change 3 to 2
5	1 2 5	change 2 to 4
6	1 4 5	change 1 to 3
7	3 4 5	change 3 to 2
8	2 4 5	change 4 to 3
9	2 3 5	change 5 to 4
10	2 3 4	Game over

Write a program that will either generate a substitution schedule based on the given team size ***n***, number of players in the field ***k***, and game duration ***t*** minutes or conclude that there is no suitable solution.

If there are several suitable schedule options, any one of them is regarded as a right solution.

Players are numbered 1 to ***n***.

Input

The input file contains three positive integers ***n***, ***k***, and ***t*** (separated by spaces).

Output

The first line of the output file must contain either a value of ***t*** if there is a suitable solution or 0 if there is no solution.

If there is a solution to the problem, the first line is followed by ***t*** lines, each one describing the composition of the team. Such lines contain ***k*** player numbers (separated by spaces). Player numbers are sorted strictly in the ascending order. Each team composition option can only be used once. Two successive lines must differ by one player only.

Limitations

$0 < k \le n \le 100$, $1 \le t \le 10^5$.

Players are numbered 1 to ***n***.

Examples

Input	Output
5 3 10	10 1 2 3 1 2 4 1 3 4 1 3 5 1 2 5 1 4 5 3 4 5 2 4 5 2 3 5 2 3 4
2 1 3	0

D. 38 Parrots

There is an iconic Russian animated cartoon about a constrictor whose length equals to 5 monkeys, 2 little elephants or 38 parrots plus one parrot wing because animals are not familiar with the metric system. Doctor Dolittle wants to check whether it is possible to find an appropriate unit of measurement for the patients waiting in queue to see him. The doctor spends most of his working hours treating the patients so he can only do the measurements during his leisure time.

When animals come to see the doctor, they wait in queue. Once a new patient joins the queue, the doctor's assistant measures his or her length and writes the result down in a list. Reception is in order of turn. Once the patient leaves, he or she is crossed out from the list. Every now and then, Doctor Dolittle checks the list and tries to find an animal that can be used to measure everyone else (not crossed out yet) so that their lengths would be positive integers.

Write a program to help Doctor Dolittle!

Input

The first line contains a value of ***n*** – the number of manipulations with the patients list. The next ***n*** lines contain one specific manipulation with the patients list. Three types of manipulations are possible:

- Enqueueing the patient to the list – a line that starts with the "+" (plus) symbol followed by the length of the new patient (without spaces)
- Dequeueing the patient from the list – a line containing a single symbol "**–**" (minus)
- Searching for a patient that could serve as a common unit of measurement – a line containing a single symbol "**?**" (question mark). This manipulation applied for nonempty list only.

Output

The output file must contain exactly one line for every search. If the list contains an animal that can be used to measure all other animals on the list, the line must start with a capital letter "***Y***" followed by the length of that animal. If there is no suitable animal on the list, the line must contain a capital letter "***N***."

Limitations

$1 \le n \le 10^5$;

The length of the patient is a positive integer not exceeding 10^9.

Examples

Input	Output
10 +4 +2 ? +3 ? - - ? +1 ?	Y2 N Y3 Y1

E. Black Box

In the course of decoding flight recorders, an error was discovered in the microcode of one of the controllers that resulted in distortion of the data recorded. According to the results of the investigation, instead of recording ***n***-digit binary numbers ($x_1x_2x_3\ldots x_n$), the device recorded ***n***-digit results of multiple summing of the initial value with a simultaneous shift to the left
($x_1x_2x_3\ldots x_n + x_2x_3\ldots x_n0 + x_3\ldots x_n00 + \ldots + x_n00\ldots0$).
As the capacity of the register where intermediate data are stored equals to ***n*** bits, most significant bits resulting from shifts and summing were discarded.
For example, initial number 0101 would be transformed into 1011:

$$(0101 + 1010 + 0100 + 1000 = 1011).$$

Moreover, the result of summing was recorded in the order from the least significant bit to the most significant one (i.e., the least significant bit on the left and the most significant, on the right), so the number from the example above would be eventually transformed into 1101.
Write a program that will restore the initial number from the encoded one and print the result in the order from the least significant bits to the most significant one.

Input

The first line of the input file contains a value of ***n*** – how many bits there are in the given number. The second line contains the summing result – a binary ***n***-bit number recorded starting from the least significant bit.

Output

The output file must contain a single line with the initial ***n***-bit number recorded starting from the least significant bit.

Limitations

$1 \le n \le 2^{1\,000\,000}$

Examples

Input	Output
4 1101	1010

F. Spying Game

It is a well-known fact that most of intelligence information comes from public sources. Following this rule, the Fairy-Tale Kingdom established an analytical unit. The first task of the unit was to map the shipment routes for magic crystals in the neighboring Magic Land. The following information was collected:

- Crystals can only be shipped from one city to another
- There is a crystal mine near one of the cities – the source of crystals
- Crystals are processed in one of the cities – and that is the final destination
- To make it harder to intercept the trucks, crystal shipment routes often change
- Only certain roads are used for crystal shipment
- Each road is used to ship crystals in one direction only
- The roads are mapped in such a way that it is impossible to return crystals to the city where they have already been, no matter which one of the cities was the origin
- Any two cities are connected by no more than one road.

The analysts managed to collect information about the number of routes between the source of crystals and other cities. Your task is to plot a possible shipments map.

There are n cities in the Magic Land, with pairs of cities connected by a road. The cities are numbered from 1 to n. Only some of roads are used to ship magic crystals. It is known that the crystal mine is located in city m. The number of possible routes from city m to other cities is represented by vector D, where $D[i]$ is the number of different routes from city m to city i.

Write a program that will plot a possible shipments map based on given data (n, m, and vector D). If there are several possible maps, any one of them is acceptable.

Input

The first line in the input file contains integers n and m separated by a space. The next line contains numbers $D[1]$, $D[2]$, ... $D[n]$ separated by spaces.

Output

The first line of the output file contains a single positive integer k – the number of roads used for crystal shipments. Each of the following k lines contains a pair of numbers – numbers of cities A_i and B_i, which means crystals can be shipped from city A_i to city B_i.

Limitations

$2 \le n \le 60$, $1 \le m \le n$, $0 \le D[i] \le 2^{62}$.

Examples

Input	Output
5 1 0 1 1 3 3	6 1 2 1 3 1 4 2 4 3 4 4 5

G. Alice and Bob

"Your passion for gambling will do you no good," Bob said as he agreed to play one more game with Alice.

The rules of the game are the following: players have a set of ***n*** positive integers. They take turns. During each move, the player (either Alice or Bob) must choose two subsequent numbers and divide each of them by they greatest common factor other than 1. If one of the players is unable to make a move, he or she loses.

The question is the following: who will eventually win if both players do their best? Do not forget that Alice always starts first.

Input

The first line contains a positive integer n ($2 \le n \le 50$) – the initial number of elements in the set.

The second line contains ***n*** positive integers separated by spaces, each of them smaller or equal to 10^9 – elements of the set.

Output

The single line must contain the name of the winner. If Alice wins, print "Alice," and if Bob does, print "Bob" (without quotation marks).

Limitations

$2 \le n \le 50$

Examples

Input	Output
3 2 4 2	Bob
4 3 9 6 18	Alice
5 630 680 735 578 510	Alice
10 49 70 80 70 70 70 70 56 98 5	Bob

H. Exam

A student named Vasily had to solve the following problem during his Unified State Exam in Computer Science.

Executor ***A17*** works with three commands numbered 1 through 3:

1. Add 1 to the current value
2. Add integer ***x*** to the current value (***x***>1)
3. Multiply the current value by 7.

The first command adds 1 to the number on the screen, the second one adds ***x***, and the third one multiplies it by 7.

Program for Executor ***A17*** contain a sequence of commands numbered 1 through 3. Executor ***A17*** transforms the number on the screen in accordance with the given sequence of commands. The first value is always 1. The final result of running the program is ***N***.

After pondering on the subject for a while, Vasily calculated the value of ***K*** – number of program options for Executor ***A17*** resulting in a certain number ***N***.

For example, when ***x*** = 5, ***N*** = 7, the number of options ***K*** = 4 (numbers of commands):

- 1, 1, 1, 1, 1, 1;
- 2, 1;
- 1, 2;
- 3.

After the exam, Vasily started wondering how to calculate the number ***x*** based on input data ***N*** and ***K***.

Write a program that will determine the smallest number ***x*** meeting the problem's requirements based on given numbers ***N*** and ***K***.

Input

The first line of the input file contains two positive integers ***N*** and ***K*** separated by a space.

Output

The output file must contain a single number ***x*** – the smallest possible number meeting the requirements, if such number exists, otherwise output file must contain 0.

Limitations

$0 < K \le 2^{60}$; $0 < N \le 10^4$; $1 < x < N$.

Examples

Input	Output
7 4	5
14 3	0

I. Hole Punch

The task is to punch ***n*** identical holes along the central axis of a narrow paper tape. The holes should be 1 cm apart and ***n*** identical holes must be punched (where ***n*** is always an even number). A hole punch always makes two holes at once, ***k*** cm apart. Let's count that it is prohibited to make one hole with the punch.
The available set of hole punches covers all possible distances between the holes, i.e. ***k***=1, ***k***=2, ***k***=3, etc. Your task is to write a program that will use the number ***n*** to list hole punches with the distance ***k*** suitable for the job.

Input

The input file contains a single positive integer ***n***.

Output

The first line of the output file contains a positive integer ***k*** – the number of suitable hole punches. The second line contains ***k*** positive integers separated by spaces and listed in the ascending order.

Limitations

$2 \le n \le 10^9$,
n – even number,
$1 \le k \le n$.

Examples

Input	Output
8	3 1 2 4

J. Multidimensional Points

The distance between points $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n)$ and $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n)$ in an $\boldsymbol{n}$-dimensional space is determined according to the Euclidean metric:
$\boldsymbol{dist}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{Sqrt}((\boldsymbol{a}_1-\boldsymbol{b}_1)^2 + (\boldsymbol{a}_2-\boldsymbol{b}_2)^2 + ... + (\boldsymbol{a}_n-\boldsymbol{b}_n)^2)$,
where ***Sqrt***() is square root function. Some optimization tasks require calculation of partial distance between points – only for coordinates with indices within a certain range $[\boldsymbol{i}, \boldsymbol{j}]$: $\boldsymbol{dist}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{i}, \boldsymbol{j}) = \boldsymbol{Sqrt}((\boldsymbol{a}_i-\boldsymbol{b}_i)^2 + (\boldsymbol{a}_{i+1}-\boldsymbol{b}_{i+1})^2 + ... + (\boldsymbol{a}_j-\boldsymbol{b}_j)^2)$. Such calculations can be performed multiple times.
Write a program that will efficiently calculate partial distance based on given points.

Input

The first line contains a single positive integer $\boldsymbol{n}$ – the number of dimensions in the space. The second and third line contain $\boldsymbol{n}$ positive integers separated by spaces – coordinates of points $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively.
The fourth line contains the number $\boldsymbol{m}$ – the number of partial distance calculations. Each of the following $\boldsymbol{m}$ lines contains two positive integers $\boldsymbol{i}$ and $\boldsymbol{j}$ separated by a space, setting the limits of the interval for the calculations.

Output

The output file consists of $\boldsymbol{m}$ lines. Each of the lines contains a real number accurate to the third decimal place – calculated partial distance. The distances must be printed in the same order as the order of intervals in the input file.

Limitations

$1 \le \boldsymbol{n} \le 10^5$,
$1 \le \boldsymbol{m} \le 10^6$,
$1 \le \boldsymbol{i} \le \boldsymbol{j} \le \boldsymbol{n}$,
$-1000 \le \boldsymbol{a}_i, \boldsymbol{b}_i \le 1000$.

Examples

Input	Output
5 1 2 3 4 5 1 1 2 1 1 1 4 5	5.000

K. Tower of Hanoi

In an effort to master their skills, participants in the programming championship regularly solve the "Tower of Hanoi" problem. All teams use the standard classical algorithm of moving, consisting of (2^N-1) moves.

```
Procedure Hanoi (X, Y, Z: char; N: integer);
Begin
 If N>0 then
    Begin
       Hanoi (X, Z, Y, N-1);
       Writeln('Disk ', N, ' from ', X, ' to ', Y);
       Hanoi (Z, Y, X, N-1)
    End
End;
```

We have the intermediate results that some teams have achieved.

All results are represented as rows of length ***N*** (the number of disks), where element ***i*** of the string is the symbol denoting the rod on which disk ***i*** is located.

For instance, let ***N*** = 4. Then the record "***BBCA***" means that the first and second disks are on rod ***B***, the third disk on rod ***C*** and the fourth on rod ***A***.

The table below shows all the 7 moves necessary to solve the Tower of Hanoi puzzle with three disks.

Move No.	Distribution of disks among the rods
0	AAA
1	BAA
2	BCA
3	CCA
4	CCB
5	ACB
6	ABB
7	BBB

Write a program that will identify the number of the team with the maximum amount of moves in the Tower of Hanoi. If there are several such teams, determine the number of the first one.

Input

The first line contains two integers: ***N*** – the number of disks and ***M*** – the number of teams that solved the puzzle.

The next ***M*** lines contain ***N*** symbols. Line ***i*** contains the result of team ***i***. Each line represents the distribution of disks among rods ***A***, ***B***, and ***C***.

Output

A single positive integer – the number of the team with the maximum amount of moves. If there are several such teams, the team with minimum number of them is displayed.

Limitations

Each line corresponds to the distribution of disks among the rods after performing move K ($0 \le K \le 2^N-1$). All teams use the standard classical algorithm of moving with the maximum of (2^N-1) moves.

The number of disks N satisfies the condition $1 \le N \le 10^3$. Number of teams: $1 \le M \le 10^3$.

Examples

Input	Output
4 7 CAAA AAAA CCCB CBAA BBAA BBCA CCCA	3
3 4 AAA BBB BAA BBB	2

L. Fence

In a country set among forests, there is a small village consisting of N houses. The people of this village are mainly engaged in animal husbandry, raising goats and sheep. The village consists of houses of different shapes and sizes. The villagers have only one problem – wolves attacking their livestock. In order to protect their property and provide security, the residents have decided to build a single fence around the village. For the residents it is very important to minimize the length of the fence, thereby reducing the cost of construction. Nevertheless, they want the fence to be at least R meters away from each house. The shape of house i is defined by a set of K_i vertices (a_j, b_j) of a polygon with real coordinates.

Write a program that will compute the minimum length of the fence.

Limitations

$1 \le N \le 1000$

$3 \le K_i \le 100\,000$

$3 \le \sum K_i \le 100\,000$

$0 \le R \le 10^3$

$-10^3 \le a_j, b_j \le 10^3$

Input

The first line of the input file contains a positive integer N – the number of houses in the village, and a real number R – the distance from the houses to the fence.

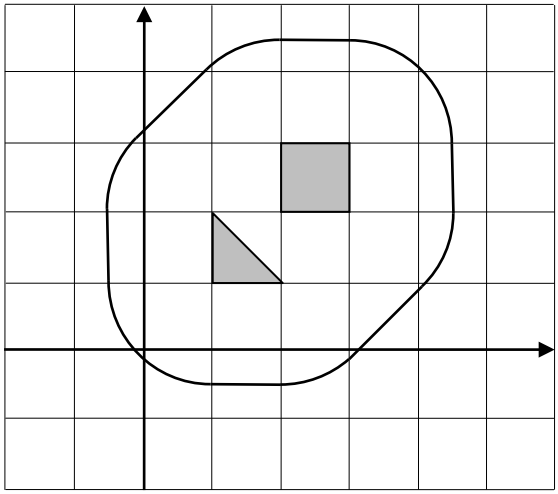
Next come N blocks describing each house. In each block, the first line is the number of vertices K_i of the polygon. The following K_i rows contain pairs of real numbers a_j and b_j, specifying the coordinates of the polygon vertices.

Output

The output file must contain a real number L accurate to five decimal places – the minimum length of the fence.

Input	Output
2 1.5 3 1 1 1 2 2 1 4 2 2 2 3 3 3 3 2	16.25321

Explanation for example



Rybinsk State Aviation Technical University

Program Committee

- Sergey G. Volchenkov,
- Vladimir N. Pinaev,
- Michael Y. Kopachev,
- Andrey Mirzoyan,
- Alexander Kiselyov,
- Dmitry Shalaev.