Problem A. Array access

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 mebibytes

During the course on compiler construction, students were assigned the task: implement parser for expressions with array access for Pascal programming language. Student Vasya has slacked on this task, and implemented only a very small language subset, consisting of a single integer constant 0 and accesses to a single two-dimensional array a. In other words, his "language" has the following grammar:

```
  expr ::= 0 | a[expr,expr]
```

After seeing this sad result, teacher assigned additional task to Vasya: suppose array a is defined as

```
a: array [0..N - 1, 0..N - 1] of 0..N
```

Given N and values of array elements, generate shortest possible expression in Vasya's language which will have the value of N.

Input

Input file contains integer N, followed by N^2 integers — values a_{ij}, in row-by-row order ($1 \leqslant N \leqslant 22$, $0 \leqslant a_{ij} \leqslant N$).

Output

Output file must contain a single string — expression in Vasya's language. Expression must exactly correspond to the grammar above. If there is no expression with the value of N, output string "`IMPOSSIBLE`". If there are several shortest expressions, output any of them.

Examples

input.txt	output.txt

2
1 2
0 0
	a[0,a[0,0]]

3
2 0 0
0 3 0
0 0 0
	IMPOSSIBLE

Problem B. Ball of fir

Input file:	`input.txt.`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Mathematicians from the Institute of New Year Research developed the mathematical model of the New Year fir-tree. In this model it is supposed that the fir is a sphere with a radius of 1 meter. Modeling of a fir garland is the most difficult.

A fir garland is modeling with a curve located on the sphere. The curve begins at the topmost point of the sphere and ends at the bottommost point of the sphere. Moreover the curve makes exactly N turns around the sphere.

Describe in more detail how the curve looks. Consider spherical coordinates φ (longitude) and ϑ (latitude). The initial value of longitude equals 0, longitude makes exactly N turns and the final value equals 0 again. The initial value of latitude equals 90 degrees, the final value equals -90 degrees. The increment of latitude is proportional to the increment of longitude.

Your program must, for given N, calculate length of the curve which models a fir garland.

Input

Input file contains a single integer N ($1 \leqslant N \leqslant 100$).

Output

Output file must contain a single real number — length of the curve (in meters) with at least 3 correct digits after decimal point.

Examples

`input.txt.`	`output.txt`
`1`	`5.270`
`3`	`12.612`

Problem C. Compression Research

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Many compression algorithms are based on finding frequently repeating substrings in the input data. Since it is often impractical to search the whole input for repetitions, only a limited *compression window* is considered on each step.

While researching a new compression algorithm, young computer scientist Vasya encountered the following problem.

Consider input string of N bits. Let the compression window to be any substing of M bits. Inside each compression window, find the maximum number of occurrences of any substring of length L ($L \leqslant M$).

Input

First line of input file contains integers L and M ($1 \leqslant M \leqslant N \leqslant 2 \cdot 10^5$, $1 \leqslant L \leqslant 100$). Second line input file contains a string of length N, each character either '`0`' or '`1`'.

Output

Output file must contain $N - M + 1$ integers — maximum substring frequencies for each compression window.

Examples

input.txt	output.txt
2 10 0101010101	5
1 3 1110000	3 2 2 3 3

Note

For example, in the first sample input compression window length is equal to the string length, so there is only a single window. Most frequent substring of length 2 is "`01`", which occurs 5 times.

Problem D. Door and wallpaper

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

Young builder Vasya was requested to hang wallpaper on the wall. The wall is a rectangle of W meters wide and H meters high, with a rectangular door of w meters wide and h meters high, located at the left side of the wall. Wallpaper is packaged into rolls. Each roll is 1 meter wide and D meters long, which Vasya may have to cut into shorter stripes of equal width.

Vasya has a high standard for work quality, so he must:

- Cover with wallpaper the entire wall except for the door, without gaps or overlapping.
- Hang wallpaper in vertical stripes of length $H - h$ over the door and H elsewhere.
- Cut all rolls into the same sequence of stripe lengths.

Your program must calculate the minimum number of wallpaper rolls required to complete the task.

In the example below it is optimal to cut rolls is into stripes of lengths $3 + 2 + 1$. If different rolls could be cut into different stripe sequences, then a better solution would be to cut one roll into lengths $3 + 3$ and another one into $2 + 2 + 2$.

Input

The input file contains integers W, H, w, h, D ($1 \leqslant W, H, w, h \leqslant 10^9$; $1 \leqslant D \leqslant 2 \times 10^9$; $h \leqslant H$; $w \leqslant W$). It is guaranteed that the answer exists for given input.

Output

Output must contain a single integer — the minimum number of rolls.

Examples

input.txt	output.txt
5 3 3 1 6	3

Problem E. Elite number

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

Let's call an integer number *elite* if it is divisible by every digit in its decimal representation.

Given integer x, your program must find the smallest elite number greater or equal than x.

Input

Input file contains integer x ($1 \leqslant x \leqslant 10^{10}$.

Output

Output file must contain a single integer – the smallest elite number.

Examples

input.txt	output.txt
10	11
7777777777	7777777777
579916	593595

Problem F. Far Eastern Federal Clock

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

Once upon a time there was a large country with many provinces and a great government. The government noticed that citizens of the furthest province are unhappy, and decided to do something for them.

After a serious sociological research, the government decided that the main problem of the province is the fact that every citizen has to buy his own watch to measure time. Thus the government decided to build a large tower with a giant analog clock on it, so that citizens could look at the common clock and save money on watches.

Many efforts and resources were spent, and finally the clock has been built and officially started in a grand ceremony. As the ceremony finished, people noticed that the clock has a small problem — it measures time incorrectly.

Since all the money allocated to this project was already spent, it was impossible to fix the clock. Instead, the government introduced a new position of Senior Clock Manager, whose responsibility was to adjust the clock by manually moving its hands.

It was decreed that:

- The clock will always be adjusted exactly at midnight.
- During the rest of the day, the clock will be adjusted periodically, every p minutes.
- The difference between the time displayed by the clock and the actual time must never exceed m minutes. Note that the smallest of all possible differences for a given hand positions is picked, for example, if the clock calculated time as 23:50 while the actual time is 00:05, the difference is 15 minutes.

Since the clock hands are very heavy, the Clock Manager's job is not an easy one. To help him, find the period of adjustment minimizing the total distance by which he must move the clock hands throughout the day.

The clock has two hands — for minutes and hours. Every minute, the minute hand jumps clockwise by t degrees, and the hour hand jumps by $t/12$ degrees. (A correct clock should have $t = 6$).

An adjustment is made immediately after the jump, and the effort is equal to the sum of angles between the current and the correct positions of both minute and hour hands.

Input

Input file contains floating point number t followed by integer m ($0 \leqslant t \leqslant 10^4$, $1 \leqslant m \leqslant 10^4$, t has no more than 3 digits after decimal point).

Output

Output file must contain minimum total effort s in degrees, with absolute error less than 10^{-2}, and the corresponding adjustment period p, $1 \leqslant p \leqslant 1440$. If there are several answers with the same total effort, output the one with maximum p.

Examples

input.txt	output.txt
12 1	9360.0000 2
18.0 90	1440.0000 30
6 1	0.0000 1440

Note

In the first sample, the clock moves twice as fast as it should, so every minute the error is increased by one minute. This requires an adjustment every two minutes.

In the second sample, the clock moves three times as fast as it should, but the acceptable error is much higher. It turns out that every 30 minutes the position of the minute hand coincides with the correct one, so if we choose the interval of 30 minutes, only the hour hand must be moved.

Problem G. Ganking

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

Popular on-line game "Attack Of The Moderns 3" is played by two teams of 5 players each. During the match, players run around the map and try to kill members of opposing team by attacking them with various weapons and magic spells. Killed players respawn after a certain period of time.

All attacks are recorded by the game for statistics gathering. Each attack is described by values t, a, v, k, where t is a time in seconds since the game start, a — number of the attacking player, v – number of the player who was attacked, $k = 1$ if this attack killed the victim and 0 otherwise. Players of the first team are numbered from 1 to 5, players of the second team are numbered from 6 to 10.

Gank is an event when one or more players attack and kill a single opponent while his teammates are elsewhere and unable to help.

Specifically: let G be a set of players who attacked the victim during last T seconds of the game before the kill. A kill is counted as a gank, if in that period of time:

- victim only attacked players from set G, and
- players from set G attacked and were attacked by only the victim.

All the players from the set G who attacked the victim of the gank are considered participants of this gank.

Your program must, given value of T and a sequence of N attack descriptions, count number of ganks each player have participated in.

Input

Input file contains integers N and T followed by N quartets of integers t_i, a_i, v_i, k ($1 \leqslant N \leqslant 10000$, $1 \leqslant t_i \leqslant t_{i+1} \leqslant 10^5$, $1 \leqslant T \leqslant 10^5$. Either $1 \leqslant a_i \leqslant 5 < v_i \leqslant 10$ or $1 \leqslant v_i \leqslant 5 < a_i \leqslant 10$). Time between sequential kills of the same victim is greater than T.

Output

Output file must contain 10 integers — number of ganks for each player.

Examples

`input.txt`	`output.txt`

`7 4`
`1 1 6 0`
`2 2 6 0`
`5 3 6 1`
`10 7 1 1`
`10 7 2 1`
`20 8 1 1`
`20 3 7 1`
	`0 1 2 0 0 0 0 1 0 0`

Problem H. Hexes in viewport

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

A game development company has started prototyping a new game, which is played on the field based on the hexagonal grid. The first task is to display a part of the grid in the player's viewport. To speed up the prototype phase, it was decided to use text-based representation instead of graphics.

Hexagonal grid is composed of nearly perfect hexagons with the side of N characters. In every hexagon, the top and the bottom sides are composed of N '`_`' characters (ASCII 95), the right-top and the left-bottom sides are composed of N '
' characters (ASCII 92), the left-top and the right-bottom sides are composed of N '`/`' characters (ASCII 47). All other characters of the field are '`.`' (ASCII 46).

The grid is assumed to be infinite, with the position $(0; 0)$ corresponding to the leftmost character of the top side of hexagon. The player's viewport is a rectangle displaying some part of the field.

You program must, given the coordinates x, y of the top left corner of viewport and w, h — the viewport width and height, output contents of the viewport.

Input

Input file contains integers N, x, y, w, h ($1 \leqslant N \leqslant 100$, $0 \leqslant x, y \leqslant 10^9$, $1 \leqslant w, h \leqslant 100$).

Output

Output file must contain h lines of w characters each — the viewport content.

Example

input.txt	output.txt
3 0 2 33 10	..../..... ...../..... .... ___/....... ___/....... ___ ...../... ..../... ..../... ..../..... ...../..... ...../.... ___/....... ___/....... ___/..... ... ./... ..../... ..../..... ...../..... .... ___/....... ___/....... ___ ...../... ..../... ..../... ..../..... ...../..... ...../....

Problem I. IT over the bridge

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	64 mebibytes

A certain large organization has an office building with extensive local area network. Recently said organization expanded to a new office building, where a new local area network was installed. Unfortunately, new building is located on the remove island, and the infrastructure there is plagued by all kinds of problems: disconnects, power outages, network packet drops etc.

Engineers of organization's IT department installed two monitoring servers, one for each building. Each server writes a log, where it stores type of failure for every problem it detects.

In theory, both logs should be identical. However, it turned out that even monitoring subsystem itself has issues: some problems are detected only by one server, servers sometimes erroneously report non-existing problems, and even timestamps are unreliable since server clocks are not synchronized.

Engineers decided to represent each log as a string of L letters, one letter for each problem. To filter out monitoring errors, they decided to consider *real problem history* to be a longest sequence of problems of the same type which occur in both logs in the same order.

While trying to improve the network, engineers frequently need to compare not only the whole logs, but selected segments of them.

Given two strings a and b representing two logs, and a sequence of N requests of the form a_1, a_2, b_1, b_2, your program must for each request output the length of real problem history produced by comparing segment from a_1 to a_2 of the first log with segment from b_1 to b_2 of the second log.

Input

First two lines of input files contain strings a and b of equal length L. Third line contains integer N. Following N lines contain 4 integers each — values $a_{i,1}a_{i,2}b_{i,1}b_{i,2}$ ($1 \leqslant L \leqslant 100$, $1 \leqslant N \leqslant 10^6$, $1 \leqslant a_{i,1} \leqslant a_{i,2} \leqslant L$, $1 \leqslant b_{i,1} \leqslant b_{i,2} \leqslant L$, strings a and b are composed of lower-case Latin letters).

Output

Output file must contain N integers — real problem history lengths.

Examples

input.txt	output.txt
abcd abcd 2 1 2 3 4 1 3 2 4	0 2
aaazaaz zzzazzz 1 4 7 1 7	3

Problem J. Jokers

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 mebibytes

Once upon a time Marfa Gennadievna bought a pack of 54 playing cards containing two jokers. She put the cards face down and chose N cards of them randomly (probability distribution is uniform).

Please calculate the probability that it will be at least one joker among chosen cards.

Input

Input file contains a single integer N ($2 \leqslant N \leqslant 54$).

Output

Output file must contain a single real number — required probability with at least 6 digits after decimal point.

Examples

input.txt	output.txt
2	0.073375
10	0.338924