# Problem A. Spaceship Connections

| | |
|---|---|
| Input file: | `connections.in` |
| Output file: | `connections.out` |
| Time limit: | 3 seconds |
| Memory limit: | 256 Megabytes |

There are $N$ Mooncats' spaceships in open space near the Moon. Ships' commanders (of course, all of them are mooncats, but it is not important in our problem) want to discuss the battle with moon rabbits which will happen soon. For this purpose, they need to establish some wireless connections between ships. But there are some limitations on them. First of all, spaceships all belong to one plane in $3D$ space. More than that, if we could look at this plane with our sight coinciding with plane's normal vector, we could see that Mooncats' ships form a regular polygon with $N$ vertices. So, there are restrictions on connections. First restriction is that no ship can be connected to more than one ship at a time. Second restriction is that if we draw line segments between pairs of ships that are connected, no two segments could intersect (in other case, both connections will be broken due to very high noise).

Mooncats' admiral wants you to implement a system which will check if a connection between ships is possible. A connection is possible if after establishing it, no restrictions will be violated. And more than that, admiral wants you to develop a real-time system which will be able to track connections and disconnections between ships.

## Input

In the first line of input file there are two integer numbers $N$ and $M$ — number of Mooncats' spaceships and number of operations to process ($1 \le N, M \le 100\,000$). In the following $M$ lines there are operations. Each operation has form `CONNECT i j` which means that commanders of ships $i$ and $j$ want to establish a connection between their ships, or `DISCONNECT i j` which means that commanders of ships $i$ and $j$ have finished negotiations and want to break the connection. Numbers $i$ and $j$ are different.

## Output

For each operation, output one character `+` if the operation was successful or `-` if the operation failed. Operation of `CONNECT` type is considered successful if there is no connection between the specified ships and it is possible to establish this connection without violating the restrictions, otherwise it is unsuccessful (in this case, the new connection is not established). Operation of type `DISCONNECT` is successful if at the moment of its execution there was a connection between specified ships, otherwise it is unsuccessful. All characters should be on a single line.

## Example

| connections.in | connections.out |
|---|---|
| 4 8<br>CONNECT 1 3<br>CONNECT 2 4<br>DISCONNECT 1 3<br>CONNECT 2 4<br>DISCONNECT 2 4<br>CONNECT 1 2<br>CONNECT 3 4<br>DISCONNECT 1 3 | +-++++- |

# Problem B. Mooncat DNA

| | |
|---|---|
| Input file: | dna.in |
| Output file: | dna.out |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Not only rabbits, but even cats live on the Moon. Mooncats are not so fast-reproducing as moon rabbits, so they evolved some other mechanisms of defending them from extinction. For instance, mooncats have a very complicated DNA structure. This structure is so complicated that scientists don't really know how to decode it.

A typical Mooncat DNA consists of up to 26 types of nucleotides and can be rather long to work with, so scientists decided to take some fragments of DNA and study their properties hoping to get key to DNA decyphering. Now they have a theory which they want to prove. Let us consider a fragment $s$ of DNA consisting of $n$ nucleotides (for simplicity, they will be denoted as small Latin letters). Scientists call some sequence $v$ of nucleotides a *subfragment* of $s$ if there is a subsequence of consecutive nucleotides in $s$ which equals to $v$. Let us call some DNA fragment $v$ *good* for fragment $u$ if there exists some DNA fragment $w$ such that $wuv = s$. Let us denote the set of fragments that are good for $u$ as $R_s(u)$. Let us call two fragments $u_1$ and $u_2$ *similar* if $R_s(u_1) = R_s(u_2)$. Clearly, such definition of similarity form an equivalence relation on subfragments of $s$. Scientists believe that study of equivalence classes of this relation will help them. Your task is to find sizes of these classes. We consider two subfragments different if they are different as strings, so they could occur more than once in a given fragment.

## Input

The first line of the input file will contain the fragment $s$ of mooncat DNA. It will consist of $n$ small Latin letters ($1 \le n \le 10^5$).

## Output

You should output sizes of similarity classes in one line, splitting them by one space. Sizes are to be output in ascending order.

## Example

| dna.in | dna.out |
|---|---|
| abacaba | 1 1 2 2 4 4 4 4 |

Comment:

In the example, we have the following eight classes of similarity: {“”}, {“a”}, {“b”, “ab”}, {“ba”, “aba”}, {“c”, “ac”, “bac”, “abac”}, {“ca”, “aca”, “baca”, “abaca”}, {“cab”, “acab”, “bacab”, “abacab”} and {“caba”, “acaba”, “bacaba”, “abacaba”}.

# Problem C. Function

| | |
|---|---|
| Input file: | `function.in` |
| Output file: | `function.out` |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Moon rabbits reproduce really fast. Not as fast as Fibonacci rabbits, but still very quickly. Mooncat scientists discovered that the quantity of moon rabbits changes according to the following law: $x_{t+1} = f(x_t)$, where $x_t$ is the quantity of moon rabbits at moment $t$. Surely, if we could know function $f$ exactly, we could compute the quantity of moon rabbits at any moment with any initial conditions. Unfortunately, mooncat scientists don't know the function itself, but they were able to state some important properties of $f$. First of all, we will consider $f$ only on positive integer values of argument (there could not be zero of negative quantity of moon rabbits). Function $f$ is strictly increasing, thus $f(x+1) > f(x)$ for every $x$. Of course, $f(x)$ is integer for every $x$ (have you ever seen 2.5 moon rabbits or $\pi$ moon rabbits?). And more than that — after countless experiments, mooncat scientists were able to discover that $f(f(x)) = 3x$. Scientists are sure that it is possible to recover function $f$ using these properties. Help them to do it.

## Input

The input file contains $M$ test cases. The first line of the input file contains the integer $M$ ($1 \le M \le 1000$). Each of the following $M$ lines describes one test case and contains one integer $N$ ($1 \le N \le 10^{18}$).

## Output

The output file should consist of $M$ lines. Each line should contain the value $f(N)$ for the respective test.

## Example

| function.in | function.out |
|---|---|
| 1<br>7 | 12 |

# Problem D. Starship Junction

| | |
|---|---|
| Input file: | junction.in |
| Output file: | junction.out |
| Time limit: | 4 seconds |
| Memory limit: | 256 Megabytes |

Spherical horses in vacuum are flying on their battlecruiser to enslave the inhabitans of the Moon. In the center of the battlecruiser there is a big junction, where a lot of corridors meet.

There is strong subordination in spherical horses' army. Each horse on the battlecruiser has a unique number. All roads are numbered in clockwise order. Horses in vacuum arrives at the junction on their segways at some time. If someone can, he starts to cross the junction. There can be only one segway passing the junction at every moment.

If there is only one segway near the junction, it starts passing the junction immediately. Let us describe which segway will start passing the junction, if there is more than one segway waiting. First of all, there should be no segway on the road to the right from segway which will start passing (for road number $i$, road to the right is road number $i + 1$, except for road $M$, for which road to the right is road 1). There is only one exception from this rule — when on every road there is a segway waiting to go (sure, there could be no road without free road to the right in this case). In this case consider all roads to satisfy the first rule. If there are several segways satisfying the first rule, a segway which came first is chosen. If there is a tie, a segway from the road with minimal number is chosen. If there is still a tie, a segway with minimal serial number is chosen.

When some horse on the segway is on the junction, it takes him some time to pass the junction. After that, the segway goes away from the junction and if there is another segway, it can start passing the junction. If a few segways arrive at junction and want to leave it simultaneously, then first all segways arrive and then the leaving segway is determined.

One day, the system controlling the junction passage broke. Please help spherical horses in vacuum to solve their big problem of controlling this junction.

## Input

In the first line of the input there will be two numbers $N, M$ — the quantity of segways to process and the number of roads ($1 \le N \le 10^5, 1 \le M \le 10^9$). Then $N$ lines follow. Each line contains four integers $x, y, z, w$ where $x$ is the serial number of the segway, $y$ is the number of the road this segway came by, $z$ is the time when the segway came and $w$ is a passing time of the junction of the segway ($1 \le x, y, z, w \le 10^9$).

## Output

For each moment of time when some segway leaves the junction, output the number of that segway and the time at which it leaves the junction. Output leaving segways in ascending order of leaving time.

## Example

| junction.in | junction.out |
|---|---|
| 3 3<br>3 1 1 10<br>2 2 1 8<br>1 3 1 6 | 3 11<br>1 17<br>2 25 |

# Problem E. LCM

| | |
|---|---|
| Input file: | `lcm.in` |
| Output file: | `lcm.out` |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Our spy satellite has intercepted a secret message transmitted by moon rabbits. Thanks to our intelligence, we already know that each secret message consists of $N$ integers. Because of encryption, we know only the least common multiple of all numbers in the message. Your task is to find out how many different messages could have been sent.

## Input

Input file contains only two integer numbers $N$ and $T$ — number of words in the message and their least common multiple ($1 \le N \le 1000$, $1 \le T \le 2 \cdot 10^9$).

## Output

Output only one integer number — answer to the problem.

## Example

| lcm.in | lcm.out |
|---|---|
| 2 6 | 9 |

# Problem F. MasterSpark

| | |
|---|---|
| Input file: | masterspark.in |
| Output file: | masterspark.out |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Shoot through the Galaxy, Master Spark!!!

Master Spark user

This was a decisive battle for future of the Earth. Invasion of the moon rabbits was inevitable and United Command decided to use the ultimate weapon called "Master Spark". It is a powerful all-annihilating beam. The destructive beam is very big, so one can assume it has a shape of a cylinder of radius $r$ (infinite in both front and back directions). Master Spark went through the moon annihilating all matter on its way. As you know, rabbits (especially moon ones) reproduce very fast. They live on the moon surface and speed of their reproduction depends on residental area (all moon surface). So United Command needs to know the area of annihilated moon surface to estimate the number of enemies. Now this is your turn to save mankind. Calculate the destoyed area of the moon. Remember, "Master Spark" is a bidirectional infinite all-annihilating beam.

## Input

First line of input file contains four numbers — radius of the moon and coordinates of its center at the time of the "Master Spark" shot. Next line contains four numbers — radius of the destructive beam and coordinates of some point on axis of the cylinder. "Master Spark" was fired from point $(0, 0, 0)$. All numbers are integers and do not exceed $10^3$ by absolute value.

## Output

Output the destroyed area of the moon surface. The answer should be accurate to at least four decimal digits.

## Examples

| masterspark.in | masterspark.out |
|---|---|
| 10 100 0 0<br>5 1 0 0 | 168.357443 |
| 10 100 0 0<br>50 0 1 0 | 0.0000 |

# Problem G. Moon Craters

| | |
|---|---|
| Input file: | `mooncraters.in` |
| Output file: | `mooncraters.out` |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Mooncats live on the Moon. It is very cold there, so they live in the Moon's craters. Every crater has its own height that counts from the sea level being found not so long ago. Each mooncat chooses some crater and in each crater can live only one mooncat. To simplify, we represent the Moon surface as a square matrix $\{h_{ij}\}$ of size $N \times N$, each crater is a cell in this matrix, and the value in a cell denotes the height of the crater in this position.

Since life is difficult, mooncats want to live in joy, happiness and pleasure. They are rather stupid, so their pleasure is determined by a special condition: for every $1 \le i \le k \le N$ and $1 \le j \le l \le N$, the following inequality is satisfied: $h_{ij} + h_{kl} \le h_{kj} + h_{il}$. Only you can help mooncats. Check if mooncats live in pleasure, hapinness... or not.

## Input

In the first line of the input file there is only one number $N$ — size of the surface of the Moon ($1 \le N \le 1000$). Next $N$ lines contain $N$ numbers each — the heights of every crater $h_{ij}$. Every height does not exceed $10^9$ by absolute value.

## Output

The first and only line of the output file must contain "`HAPPY`" if mooncats live in pleasure and "`UNHAPPY`" otherwise.

## Examples

| mooncraters.in | mooncraters.out |
|---|---|
| 2<br>1 2<br>3 4 | HAPPY |
| 3<br>1 2 3<br>4 0 5<br>6 7 8 | UNHAPPY |

# Problem H. Pug on the Moon

| | |
|---|---|
| Input file: | `moonpug.in` |
| Output file: | `moonpug.out` |
| Time limit: | 10 seconds |
| Memory limit: | 256 Megabytes |

There is one unique moonpug that lives on the moon. Mooncats know about this. They decided to send a satellite to the space to observe the moonpug. Moonpug is informed about it. And it is not interested in disclosure of it's location. Moonpug knows that a satellite can't observe moonpug if it is located inside a lunar crater. Moonpug can run with a constant speed of one meter per second. And it needs to get from point $A$ to point $B$ in a way that minimizes time when mooncat's satellite observes the moonpug.

## Input

In the first line of input there is one integer number $N$ — the number of lunar craters ($0 \le N \le 100$). Each crater is a non-degenerate convex polygon that is described as an array of vertices in clockwise or counterclockwise order. In the next lines there are descriptions of craters. Description of one crater starts with an integer $M_i$ on a line by itself — number of vertices in the polygon ($3 \le M_i \le 1000$). Next $M_i$ lines contain two integers $X_{i,j}$ and $Y_{i,j}$ each — the coordinates of $j$-th point in the $i$-th crater. In the last line of input there are four integers — coordinates of points $A$ and $B$. All coordinates are intergers not greater than $10^4$ by absolute value.

## Output

Write to the output file only one number with at least six digits after decimal point — the minimal time that satelline will observe the moonpug while it is travelling from point $A$ to point $B$.

## Example

| moonpug.in | moonpug.out |
|---|---|
| 1<br>3<br>0 0<br>100 0<br>50 -50<br>0 1 100 1 | 2.000000 |

# Problem I. Building Spaceships

| | |
|---|---|
| Input file: | spaceships.in |
| Output file: | spaceships.out |
| Time limit: | 20 seconds |
| Memory limit: | 256 Megabytes |

Mooncats have recently made a breakthrough in spaceship building and now want to build spaceships. They live in 3D-space like ordinary people. Mooncats are able to build spaceships only using special metal blocks. They can go from one block of a spaceship to another if they have an adjacent side. Each spaceship is a set of connected blocks, i.e. a mooncat can go from any block to any other block. Now they want to know all types of spaceships they can build, and want to build all spaceships such that each one consists of $N$ metal blocks. Two spaceships are the same if they will coincide after some moves and rotations in 3D-space.

## Input

There is only one number $N$ — number of blocks each spaceship should consist of ($1 \le N \le 9$).

## Output

On the first line of output file print $M$ — the number of different spaceships. Next lines must contain descriptions of types of spaceships. Each description consists of $N$ coordinates of blocks of spaceships in 3D-space. Coordinates must be non-negative integers not exceeding $N$. And for uniqueness, you must print the *lexicographically smallest* description of each spaceship. Description $A$ of a spaceship is lexicographically smaller than description $B$ if it satisfies the following condition: there is an integer $i$ ($0 \le i < N$) such that for each $1 \le j \le i$, coordinates of $j$-th block of $A$ and $B$ are the same, and coordinates of $(i+1)$-th block in the description $A$ are smaller than coordinates of that block in $B$. Separate the consecutive numbers in output by spaces and/or blank lines. The coordinates are compared as sequences of three numbers. You can output descriptions in any order.

## Examples

| spaceships.in | spaceships.out |
|---|---|
| 2 | 1<br>0 0 0<br>0 0 1 |
| 3 | 2<br>0 0 0<br>0 0 1<br>0 0 2<br><br>0 0 0<br>0 0 1<br>0 1 0 |

# Problem J. Starship

| | |
|---|---|
| Input file: | `starship.in` |
| Output file: | `starship.out` |
| Time limit: | 1 second |
| Memory limit: | 256 Megabytes |

Moon rabbits are planning massive colonization of Omega Prime. For that purpose, a great starship was built. But this starship needs fuel to operate. Thus moon rabbits need $F$ barrels of fuel. There are $N$ fuel providing companies on the Moon. Each company has $f_i$ barrels of fuel in their stocks. The $i$-th company provides fuel at cost $c_i$ money units per barrel ($1 \le i \le N$). Now, colonization government needs to minimize the total cost of all fuel. You are to help them with their problem and get rid of the evil moon rabbits once and forever: from Omega Prime, they will not disturb the Earth.

## Input

To minimize the risk of enemy intrusion in government plans, the moon rabbits sent us encrypted data. First line of input file contains two integer numbers $N$ and $F$ — number of fuel providing companies and total amount of fuel needed for colonization ($0 \le N \le 10^7$, $0 \le F \le 10^{18}$). Second line contains four integer numbers $f_1$, $c_1$, $a$ and $b$. To decrypt the message, you have to use formulas $f_i = (f_{i-1} \cdot a + b) \bmod 2^7$ and $c_i = (c_{i-1} \cdot a + b) \bmod 2^{31}$ for all $1 < i \le N$. $a$ and $b$ fit in signed 32-bit integer number.

## Output

Output only one integer — the total amount of money units needed to buy the necessary amount of fuel. If the total amount of fuel available on the Moon is smaller than $F$, output the total cost of all fuel on the Moon.

## Example

| starship.in | starship.out |
|---|---|
| 5 15<br>2 1 1 2 | 65 |

# Problem K. Tickets

| | |
|---|---|
| Input file: | `tickets.in` |
| Output file: | `tickets.out` |
| Time limit: | 2 seconds |
| Memory limit: | 256 Megabytes |

Mooncats are very strange creatures. Nevertheless, their understanding of luck is the same as ours. For example, they consider a ticket for moon monorail lucky if the sum of the first $N$ digits of the ticket is equal to the sum of the last $N$ (yes, tickets have $2N$ digits in their number, leading zeroes are allowed).

One day, three mooncats were going by monorail to participate in their regional MCPC (Mooncat Collegiate Programming Contest). They think that they will have luck on this contest if at least one ticket that they will buy is lucky. Considering all triples of consecutive tickets equiprobable, compute the probability that mooncats will have luck.

## Input

Input file consist of only one integer number $N$ — half the length of moon monorail tickets ($1 \le N \le 100$).

## Output

Output only one number — the desired probability. Your answer will be considered correct if it will have at least six correct digits after decimal point.

## Example

| tickets.in | tickets.out |
|---|---|
| 1 | 0.26 |