Problem A
Self-Assembly
Time Limit: 3 seconds

Automatic Chemical Manufacturing is experimenting with a process called self-assembly. In this process, molecules with natural affinity for each other are mixed together in a solution and allowed to spontaneously assemble themselves into larger structures. But there is one problem: sometimes molecules assemble themselves into a structure of unbounded size, which gums up the machinery.

You must write a program to decide whether a given collection of molecules can be assembled into a structure of unbounded size. You should make two simplifying assumptions: 1) the problem is restricted to two dimensions, and 2) each molecule in the collection is represented as a square. The four edges of the square represent the surfaces on which the molecule can connect to other compatible molecules.

In each test case, you will be given a set of molecule descriptions. Each type of molecule is described by four two-character *connector labels* that indicate how its edges can connect to the edges of other molecules. There are two types of connector labels:

- An uppercase letter ($A, \ldots, Z$) followed by $+$ or $-$. Two edges are compatible if their labels have the same letter but different signs. For example, $A+$ is compatible with $A-$ but is not compatible with $A+$ or $B-$.
- Two zero digits 00. An edge with this label is not compatible with any edge (not even with another edge labeled 00).

Assume there is an unlimited supply of molecules of each type, which may be rotated and reflected. As the molecules assemble themselves into larger structures, the edges of two molecules may be adjacent to each other only if they are compatible. It is permitted for an edge, regardless of its connector label, to be connected to nothing (no adjacent molecule on that edge).

Figure A.1 shows an example of three molecule types and a structure of bounded size that can be assembled from them (other bounded structures are also possible with this set of molecules).

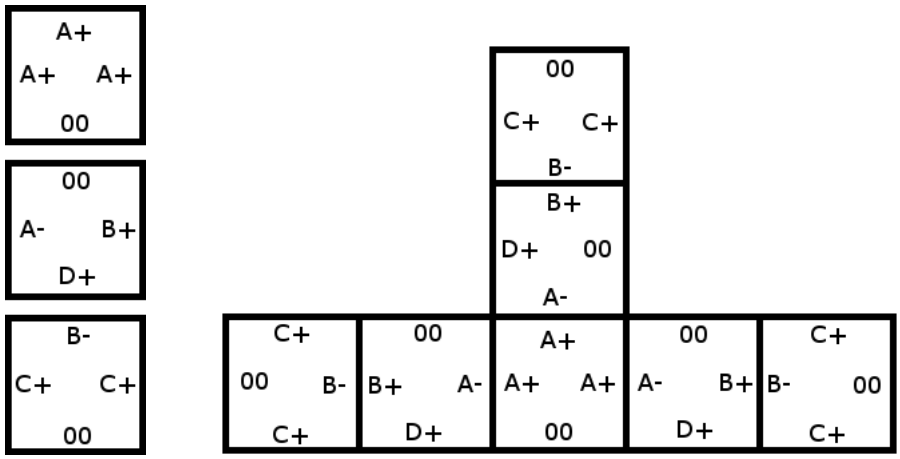


Figure A.1: Illustration of Sample Input 1.

Input

The input consists of a single test case. A test case consists of two lines. The first contains an integer n ($1 \leq n \leq 40\,000$) indicating the number of molecule types. The second line contains n eight-character strings, each describing a single type of molecule, separated by single spaces. Each string consists of four two-character connector labels representing the four edges of the molecule in clockwise order.

Output

Display the word `unbounded` if the set of molecule types can generate a structure of unbounded size. Otherwise, display the word `bounded`.

Sample Input 1	Sample Output 1
3 A+00A+A+ 00B+D+A- B-C+00C+	bounded

Sample Input 2	Sample Output 2
1 K+K-Q+Q-	unbounded

Problem B
Hey, Better Bettor
Time Limit: 4 seconds

> "*In the casino, the cardinal rule is to keep them playing and to keep them coming back. The longer they play, the more they lose, and in the end, we get it all.*"
>
> (from the 1995 film *Casino*)

Recent recessions have not been kind to entertainment venues, including the gambling industry. Competition is fierce among casinos to attract players with lots of money, and some have begun to offer especially sweet deals. One casino is offering the following: you can gamble as much as you want at the casino. After you are finished, if you are down by any amount from when you started, the casino will refund $x\%$ of your losses to you. Obviously, if you are ahead, you can keep all of your winnings. There is no time limit or money limit on this offer, but you can redeem it only once.

For simplicity, assume all bets cost 1 dollar and pay out 2 dollars. Now suppose x is 20. If you make 10 bets in total before quitting and only 3 of them pay out, your total loss is 3.2 dollars. If 6 of them pay out, you have gained 2 dollars.

Given x and the percentage probability p of winning any individual bet, write a program to determine the maximum expected profit you can make from betting at this casino, using any gambling strategy.

Input

The input consists of a single test case. A test case consists of the refund percentage x ($0 \leq x < 100$) followed by the winning probability percentage p ($0 \leq p < 50$). Both x and p have at most two digits after the decimal point.

Output

Display the maximum expected profit with an absolute error of at most 10^{-3}.

Sample Input 1	Sample Output 1
0 49.9	0.0

Sample Input 2	Sample Output 2
50 49.85	7.10178453

This page is intentionally left blank.

Problem C
Surely You Congest
Time Limit: 10 seconds

You are in charge of designing an advanced centralized traffic management system for smart cars. The goal is to use global information to instruct morning commuters, who must drive downtown from the suburbs, how best to get to the city center while avoiding traffic jams.

Unfortunately, since commuters know the city and are selfish, you cannot simply tell them to travel routes that take longer than normal (otherwise they will just ignore your directions). You can only convince them to change to different routes that are equally fast.

The city's network of roads consists of intersections that are connected by bidirectional roads of various travel times. Each commuter starts at some intersection, which may vary from commuter to commuter. All commuters end their journeys at the same place, which is downtown at intersection 1. If two commuters attempt to start travelling along the same road in the same direction at the same time, there will be congestion; you must avoid this. However, it is fine if two commuters pass through the same intersection simultaneously or if they take the same road starting at different times.

Determine the maximum number of commuters who can drive downtown without congestion, subject to all commuters starting their journeys at exactly the same time and without any of them taking a suboptimal route.

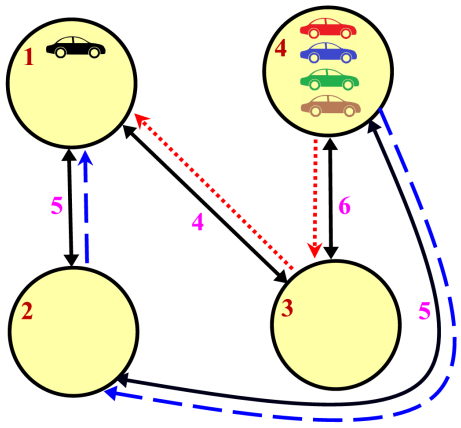


Figure C.1: Illustration of Sample Input 2.

In Figure C.1, cars are shown in their original locations. One car is already downtown. Of the cars at intersection 4, one can go along the dotted route through intersection 3, and another along the dashed route through intersection 2. But the remaining two cars cannot reach downtown while avoiding congestion. So a maximum of 3 cars can reach downtown with no congestion.

Input

The input consists of a single test case. The first line contains three integers n, m, and c, where n ($1 \leq n \leq 25\,000$) is the number of intersections, m ($0 \leq m \leq 50\,000$) is the number of roads, and c ($0 \leq c \leq 1\,000$) is the number of commuters. Each of the next m lines contains three integers x_i, y_i, and t_i describing one road, where x_i and y_i ($1 \leq x_i, y_i \leq n$) are the distinct intersections the road connects, and t_i ($1 \leq t_i \leq 10\,000$) is the time it takes to travel along that road in either direction. You may assume

that downtown is reachable from every intersection. The last line contains c integers listing the starting intersections of the commuters.

Output

Display the maximum number of commuters who can reach downtown without congestion.

Sample Input 1	Sample Output 1
3 3 2 1 2 42 2 3 1 2 3 1 2 3	2

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

Problem D
Factors
Time Limit: 2 seconds

The fundamental theorem of arithmetic states that every integer greater than 1 can be uniquely represented as a product of one or more primes. While unique, several arrangements of the prime factors may be possible. For example:

$$\begin{aligned} 10 &= 2 \cdot 5 \\ &= 5 \cdot 2 \end{aligned} \qquad \begin{aligned} 20 &= 2 \cdot 2 \cdot 5 \\ &= 2 \cdot 5 \cdot 2 \\ &= 5 \cdot 2 \cdot 2 \end{aligned}$$

Let $f(k)$ be the number of different arrangements of the prime factors of k. So $f(10) = 2$ and $f(20) = 3$.

Given a positive number n, there always exists at least one number k such that $f(k) = n$. We want to know the smallest such k.

Input

The input consists of at most 1 000 test cases, each on a separate line. Each test case is a positive integer $n < 2^{63}$.

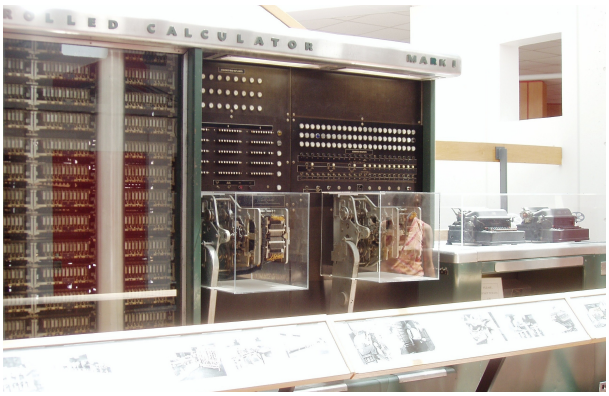
Output

For each test case, display its number n and the smallest number $k > 1$ such that $f(k) = n$. The numbers in the input are chosen such that $k < 2^{63}$.

Sample Input 1	Sample Output 1
1 2 3 105	1 2 2 6 3 12 105 720

This page is intentionally left blank.

Problem E
Harvard
Time Limit: 10 seconds



Picture from Wikimedia Commons

The term "Harvard architecture" applies to a computer that has physically separate memories for instructions and data. The term originated with the Harvard Mark I computer, delivered by IBM in 1944, which used paper tape for instructions and relays for data.

Some modern microcontrollers use the Harvard architecture – but not paper tape and relays! Data memory is organized in banks, each containing the same number of data items. Each data-referencing instruction has a byte offset f to a bank, and a bit a that is used to select the bank to be referenced. If a is 0, then bank 0 is referenced. If a is 1, then the value in a *bank select register* (BSR) identifies the bank to be used. Assume each instruction takes the same time to execute, and there is an instruction that can set the BSR's value.

For example, suppose there are 4 banks of 8 bytes each. To access location 5, either use a single instruction with $a = 0$ and $f = 5$, or set the BSR to 0 in one instruction and then use an instruction with $a = 1$ and $f = 5$. The first approach is faster since it does not require setting the BSR.

Now suppose (with the same memory) the location to access is 20. Only one approach will work here: execute an instruction that sets the BSR to 2 (unless the BSR already has the value 2) and then use an instruction with $a = 1$ and $f = 4$.

A *program* is a sequence of operations. Each operation is either

- a variable reference, written as `V`i, where i is a positive integer, or
- a repetition, written as `R`n `<program> E`, where n is a positive integer and `<program>` is an arbitrary program. This operation is equivalent to n sequential occurrences of `<program>`.

Your problem is to determine the minimum running time of programs. In particular, given the number and size of the memory banks and a program to be executed, find the minimum number of instructions (which reference memory location and possibly set the BSR) that must be executed to run the program. To do this you must identify a mapping of variables to memory banks that yields the smallest execution time, and report that execution time – that is, the number of memory references and BSR register settings required. The BSR's value is initially undefined, and changes only when an instruction explicitly sets its value.

Input

The input consists of a single test case. A test case consists of two lines. The first line contains two integers b and s, where $1 \leq b \leq 13$ is the number of memory banks and $1 \leq s \leq 13$ is the number of variables that can be stored in each memory bank. The second line contains a non-empty program with at most 1 000 space-separated elements (each `R`n, `V`i, and `E` counts as one element).

You may assume the following:

- In a repetition `R`n, the number of repetitions satisfies $1 \leq n \leq 10^6$.
- In a loop operation `R`n `<program> E`, the loop body `<program>` is not empty.
- In a variable reference `V`i, the variable index satisfies $1 \leq i \leq \min(b \cdot s, 13)$.
- The total number of variable references performed by an execution of the program is at most 10^{12}.

Output

Display the minimum number of instructions that must be executed to complete the program.

Sample Input 1

```
1 2
V1 V2 V1 V1 V2
```

Sample Output 1

```
5
```

Sample Input 2

```
2 1
V1 V2 V1 V1 V2
```

Sample Output 2

```
6
```

Sample Input 3

```
1 2
R10 V1 V2 V1 E
```

Sample Output 3

```
30
```

Sample Input 4

```
4 1
V1 R2 V2 V4 R2 V1 E V3 E
```

Sample Output 4

```
17
```

Problem F
Low Power
Time Limit: 4 seconds

You are building advanced chips for machines. Making the chips is easy, but the power supply turns out to be an issue since the available batteries have varied power outputs.

Consider the problem of n machines, each with two chips, where each chip is powered by k batteries. Surprisingly, it does not matter how much power each chip gets, but a machine works best when its two chips have power outputs as close as possible. The power output of a chip is simply the smallest power output of its k batteries.

You have a stockpile of $2nk$ batteries that you want to assign to the chips. It might not be possible to allocate the batteries so that in every machine both chips have equal power outputs, but you want to allocate them so that the differences are as small as possible. To be precise, you want to tell your customers that in all machines the difference of power outputs of the two chips is at most d, and you want to make d as small as possible. To do this you must determine an optimal allocation of the batteries to the machines.

Consider Sample Input 1. There are 2 machines, each requiring 3 batteries per chip, and a supply of batteries with power outputs $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$. You can, for instance, assign the batteries with power outputs $1, 3, 5$ to one chip, those with power $2, 4, 12$ to the other chip of the same machine, those with power $6, 8, 9$ to the third chip, and those with power $7, 10, 11$ to the fourth. The power outputs of the chips are $1, 2, 6$, and 7, respectively, and the difference between power outputs is 1 in both machines. Note that there are many other ways to achieve this result.

Input

The input consists of a single test case. A test case consists of two lines. The first line contains two positive integers: the number of machines n and the number of batteries per chip k ($2nk \leq 10^6$). The second line contains $2nk$ integers p_i specifying the power outputs of the batteries ($1 \leq p_i \leq 10^9$).

Output

Display the smallest number d such that you can allocate the batteries so that the difference of power outputs of the two chips in each machine is at most d.

Sample Input 1	Sample Output 1
2 3 1 2 3 4 5 6 7 8 9 10 11 12	1

Sample Input 2	Sample Output 2
2 2 3 1 3 3 3 3 3 3	2

This page is intentionally left blank.

Problem G
Map Tiles
Time Limit: 20 seconds

Publishing maps is not an easy task. First you need some appropriate transformation to display the earth's spherical shape in a two-dimensional plane. Then another issue arises – most high-quality maps are too large to be printed on a single page of paper. To cope with that, map publishers often split maps into several rectangular tiles, and print each tile on one page. In this problem, you will examine this "tiling" process.

The International Cartographic Publishing Company (ICPC) needs to cut their printing costs by minimizing the number of tiles used for their maps. Even with a fixed tile size (determined by the page size) and map scale, you can still optimize the situation by adjusting the tile grid.

The left side of Figure G.1 shows 14 map tiles covering a region. The right side shows how you can cover the same region with only 10 tiles, without changing the tile sizes or orientation.

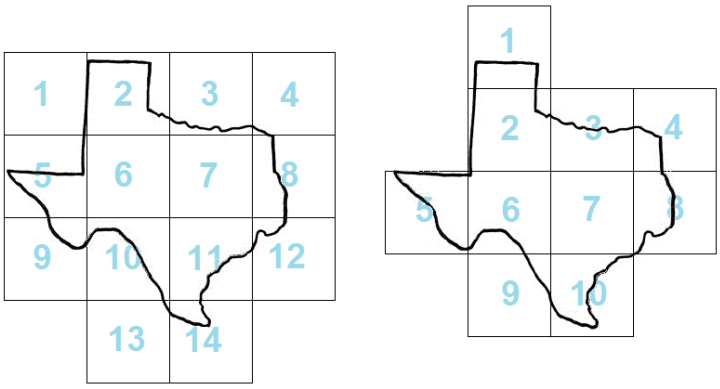


Figure G.1: Two possible ways of tiling Texas.

Your task is to help the ICPC find the minimum number of tiles needed to cover a given region. For simplicity, the region will be given as a closed polygon that does not intersect itself.

Note that the tiles must be part of a rectangular grid aligned with the x-axis and y-axis. That is, they touch each other only with their whole sides and cannot be rotated. Also note that although all input coordinates are integers, tiles may be located at non-integer coordinates.

The polygon may touch the edges of marginal lines (as in Sample Input 2). However, to avoid floating-point issues, you may assume the optimal answer will not change even if the polygon is allowed to go outside the map tiles by a distance of 10^{-6}.

Input

The input consists of a single test case. The first line of a test case contains three integers: n, x_s, and y_s. The number of polygon vertices is n ($3 \leq n \leq 50$), and x_s and y_s ($1 \leq x_s, y_s \leq 100$) are the dimensions of each tile. Each of the next n lines contains two integers x and y ($0 \leq x \leq 10x_s$, $0 \leq y \leq 10y_s$), specifying the vertices of the polygon representing the region (in either clockwise or counter-clockwise order).

Output

Display the minimal number of tiles necessary to cover the whole interior of the polygon.

Sample Input 1	Sample Output 1
12 9 9 1 8 1 16 6 16 9 29 19 31 23 24 30 23 29 18 20 12 22 8 14 0 14 8	10

Sample Input 2	Sample Output 2
4 5 7 10 10 15 10 15 17 10 17	1

Problem H

Матрёшка

Time Limit: 5 seconds

Matryoshkas are sets of traditional Russian wooden dolls of decreasing size placed one inside the other. A matryoshka doll can be opened to reveal a smaller figure of the same sort inside, which has, in turn, another figure inside, and so on.

The Russian Matryoshka Museum recently exhibited a collection of similarly designed matryoshka sets, differing only in the number of nested dolls in each set. Unfortunately, some over-zealous (and obviously unsupervised) children separated these sets, placing all the individual dolls in a row. There are n dolls in the row, each with an integer size. You need to reassemble the matryoshka sets, knowing neither the number of sets nor the number of dolls in each set. You know only that every complete set consists of dolls with consecutive sizes from 1 to some number m, which may vary between the different sets.

Picture from Wikimedia Commons

When reassembling the sets, you must follow these rules:

- You can put a doll or a nested group of dolls only inside a larger doll.
- You can combine two groups of dolls only if they are adjacent in the row.
- Once a doll becomes a member of a group, it cannot be transferred to another group or permanently separated from the group. It can be temporarily separated only when combining two groups.

Your time is valuable, and you want to do this reassembly process as quickly as possible. The only time-consuming part of this task is opening and subsequently closing a doll, so you want to minimize how often you do this. For example, the minimum number of openings (and subsequent closings) when combining group [1, 2, 6] with the group [4] is two, since you have to open the dolls with sizes 6 and 4. When combining group [1, 2, 5] with the group [3, 4], you need to perform three openings.

Write a program to calculate the minimum number of openings required to combine all disassembled matryoshka sets.

Input

The input consists of a single test case. A test case consists of two lines. The first line contains one integer n ($1 \leq n \leq 500$) representing the number of individual dolls in the row. The second line contains n positive integers specifying the sizes of the dolls in the order they appear in the row. Each size is between 1 and 500 inclusive.

Output

Display the minimum number of openings required when reassembling the matryoshka sets. If reassembling cannot be done (some of the kids might have been excessively zealous and taken some dolls), display the word `impossible`.

Sample Input 1	Sample Output 1
7 1 2 1 2 4 3 3	impossible

Sample Input 2	Sample Output 2
7 1 2 3 2 4 1 3	7

Problem I
Pirate Chest
Time Limit: 15 seconds

Pirate Dick finally had enough of fighting, marauding, theft, and making life miserable for many on the open seas. So he decided to retire, and he found the perfect island to spend the rest of his days on, provided he does not run out of money. He has plenty of gold coins now, and he wants to store them in a chest (he is a pirate after all). Dick can construct a rectangular chest with integer dimensions of any size up to a specified maximum size for the top but with an arbitrary integer height. Now he needs a place to hide the chest. While exploring the island, he found the perfect solution.

Dick will hide his chest by submerging it in a murky pond. The pond has a rectangular surface, and it completely fills the bottom of a valley that has high vertical rocky walls. Dick surveyed the pond and knows its depth for each of the squares of a Cartesian coordinate grid system placed on the pond surface. When Dick submerges the chest, it will sink as far as possible until it touches the bottom. The top of the chest will remain parallel to the pond's surface and the chest will be aligned with the grid squares. The water displaced by the submerged chest will raise the level of the pond's surface (this will occur even if there is no space around the chest for the displaced water to rise). The walls of the valley are high enough that the water can never splash out of the valley. Of course, since the chest must be invisible, its top must be strictly below the surface of the pond. Your job is to find the volume of the largest chest that Pirate Dick can hide this way.

In Figure I.1, the leftmost image shows a pond, the middle image shows a possible placement of a chest of volume 3, and the rightmost image shows a placement of a chest of volume 4, which is the maximum possible volume. Note that if the second chest were made one unit taller, its top would be visible because it would be at exactly the same height as the surface of the water.

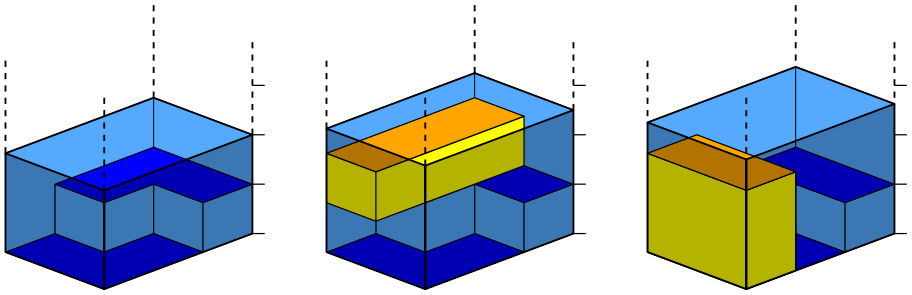


Figure I.1: Illustration of Sample Input 1.

Input

The input consists of a single test case. A test case starts with a line containing four integers a, b, m, and n ($1 \leq a, b, m, n \leq 500$). The pond's surface dimensions are $m \times n$ and the maximum size of the top (and bottom) of the chest is $a \times b$. In addition, a and b are small enough that it is not possible to cover the entire pond with a chest with top size $a \times b$. Each of the remaining m lines in a test case contains n integers $d_{i,j}$ specifying the pond's depth at grid square (i, j), where $0 \leq d_{i,j} \leq 10^9$ for each $1 \leq i \leq m$ and $1 \leq j \leq n$.

Output

Display the maximum volume of a rectangular chest with integer dimensions (where one of the dimensions of the top is bounded by a and the other is bounded by b) that can be completely submerged below the surface of the pond. If no chest can be hidden in the pond, display `0`.

Sample Input 1	Sample Output 1
3 1 2 3 2 1 1 2 2 1	4

Sample Input 2	Sample Output 2
4 1 1 5 2 0 2 2 2	12

Sample Input 3	Sample Output 3
2 3 3 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	18

Problem J
Pollution Solution
Time Limit: 1 second

As an employee of Aqueous Contaminate Management, you must monitor the pollution that gets dumped (sometimes accidentally, sometimes purposefully) into rivers, lakes and oceans. One of your jobs is to measure the impact of the pollution on various ecosystems in the water such as coral reefs, spawning grounds, and so on.

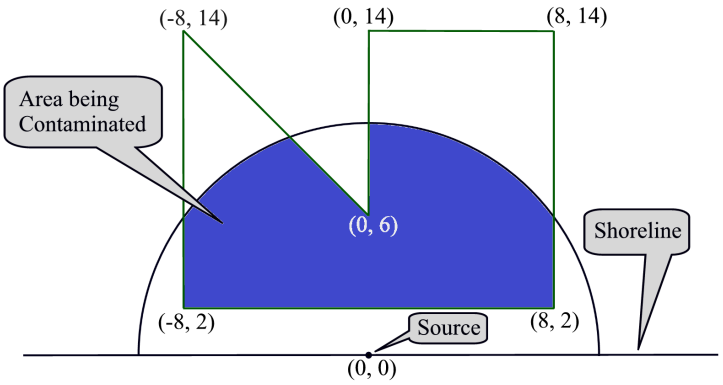


Figure J.1: Illustration of Sample Input 1.

The model you use in your analysis is illustrated in Figure J.1. The shoreline (the horizontal line in the figure) lies on the x-axis with the source of the pollution located at the origin (0,0). The spread of the pollution into the water is represented by the semicircle, and the polygon represents the ecosystem of concern. You must determine the area of the ecosystem that is contaminated, represented by the dark blue region in the figure.

Input

The input consists of a single test case. A test case starts with a line containing two integers n and r, where $3 \leq n \leq 100$ is the number of vertices in the polygon and $1 \leq r \leq 1\,000$ is the radius of the pollution field. This is followed by n lines, each containing two integers x_i, y_i, giving the coordinates of the polygon vertices in counter-clockwise order, where $-1\,500 \leq x_i \leq 1\,500$ and $0 \leq y_i \leq 1\,500$. The polygon does not self-intersect or touch itself. No vertex lies on the circle boundary.

Output

Display the area of the polygon that falls within the semicircle centered at the origin with radius r. Give the result with an absolute error of at most 10^{-3}.

Sample Input 1	Sample Output 1
6 10 -8 2 8 2 8 14 0 14 0 6 -8 14	101.576437872

Problem K
Up a Tree
Time Limit: 6 seconds

Anatoly Cheng McDougal is a typical student in many ways. Whenever possible he tries to cut and paste code instead of writing it from scratch. Unavoidably this approach causes him problems. For example, when he first learned about preorder, inorder and postorder traversals of trees, and was given code for a preorder print of a tree (shown on the left below), he simply cut and pasted the code, then moved the print statement to the correct location and renamed the procedure. However, he forgot to rename the procedure calls inside the code, resulting in the defective inorder print and postorder print code shown below.

```
void prePrint(TNode t)
{
  output(t.value);
  if (t.left != null)
    prePrint(t.left);
  if (t.right != null)
    prePrint(t.right);
}
```

```
void inPrint(TNode t)
{
  if (t.left != null)
    prePrint(t.left);
  output(t.value);
  if (t.right != null)
    prePrint(t.right);
}
```

```
void postPrint(TNode t)
{
  if (t.left != null)
    prePrint(t.left);
  if (t.right != null)
    prePrint(t.right);
  output(t.value);
}
```

At this point, Anatoly did not behave like a typical student. He actually tested his code! Unfortunately, when the results were not correct, he reverted back to typical student behavior. He panicked and started randomly changing calls in all three procedures, hoping to get things right. Needless to say, the situation became even worse now than when he started.

Anatoly's professor tested the code on a random tree of characters. When she looked at the output of his three print routines, she correctly guessed what had happened. However, instead of going directly to his code, she decided to try to reconstruct Anatoly's code just by observing the output. In order to do this, she correctly made the following assumptions:

1. The output statement in each print routine is in the correct location (for example, between the two recursive calls in the `inPrint` routine).
2. Among the six recursive calls made by the three routines, exactly two calls are to `prePrint`, exactly two are to `inPrint`, and exactly two are to `postPrint`, though potentially in the wrong routines.

Soon the professor realized that reconstructing Anatoly's code and the test tree from his output was not a simple task and that the result might be ambiguous. You will have to help her find all possible reconstructions of Anatoly's code. In addition, for each such reconstruction, you are to find the alphabetically first tree (as described in the output section) giving the observed output.

Input

The input consists of a single test case. A test case consists of three strings on three separate lines: the observed output of Anatoly's `prePrint`, `inPrint` and `postPrint` routines (in that order) on some test tree. Each of these strings consists of n uppercase letters ($4 \leq n \leq 26$), with no repeated letters in any string. The test case is guaranteed to have at least one solution.

Output

Display all possible reconstructions for the test case, ordered as described in the last paragraph below. The output for each reconstruction consists of two parts. The first part is a single line and describes the six calls in Anatoly's routines: first the two (recursive) calls in Anatoly's `prePrint` routine, followed by the calls in his `inPrint` routine, and finally the calls in his `postPrint` routine. The calls are described by the words `Pre`, `In`, and `Post`, separated by spaces. For example, if Anatoly's routines were correct, the resulting output of the first part of the reconstruction would be `Pre Pre In In Post Post`.

The second part consists of three lines and describes the first test tree that could have generated the observed outputs. The first line is the *correct* preorder print of the tree, and the second and third lines contain the correct inorder and postorder prints, respectively. The first tree is the one with the alphabetically first preorder print. If there are multiple such trees, the first of these is the one with the alphabetically first inorder print.

Every reconstruction is a sequence of 6 tokens chosen from `Pre`, `In`, and `Post`. The ordering of reconstructions is lexicographic with respect to the following ordering of tokens: `Pre < In < Post`.

Sample Input 1	Sample Output 1
HFBIGEDCJA BIGEDCJFAH BIGEDCJFAH	Pre Post In Post In Pre HFBJCDEGIA BIGEDCJFAH IGEDCJBAFH

Sample Input 2	Sample Output 2
BNLFAGHPEDOCMJIK NLBGAPHCODEIJMKF NLFAGHPEDOCMJIKB	In Pre In Post Post Pre BLNFKMEHAGPCODIJ NLBAGHPEODCMIJKF NLGAPHDOCEJIMKFB Post Pre In In Post Pre BLNFKICPGAHEODMJ NLBGAPHCODEIJMKF NLAGHPDOECJMIKFB