Problems Overview

Problem	**SQUARES**	**CARDS**	**CONTRACT**
Program name	SQUARES.EXE	CARDS.EXE	CONTRACT.EXE
Source name	SQUARES.PAS SQUARES.C SQUARES.CPP	CARDS.PAS CARDS.C CARDS.CPP	CONTRACT.PAS CONTRACT.C CONTRACT.CPP
Input file	SQUARES.IN	CARDS.IN	CONTRACT.IN
Output file	SQUARES.OUT	CARDS.OUT	CONTRACT.OUT
Time limit per test	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10
Points per test	3	3	4
Total points	**30**	**30**	**40**

The maximum total score for Round I is 100 points.

SQUARES

Problem description

We are given N squares in the co-ordinate plane whose sides are parallel to the co-ordinate axes. All the corners have integer co-ordinates and the squares do not touch or overlap.
You are required to count the number of squares visible from the origin point O, O = (0, 0).
A square is **visible** from the origin point O if there are two distinct points A and B on one of its sides such that the interior of the triangle OAB has no common points with any of the remaining squares.

Input data

The first line of the input file **SQUARES.IN** contains the integer N, $1 \le N \le 1000$, the number of squares. Each of the following N lines describes a square by specifying integers X, Y and L separated by single blank characters, $1 \le X, Y, L \le 10000$. X and Y are co-ordinates of the lower left corner (the corner with the least X and Y co-ordinates) and L is the side length.

Output data

The first and the only line of the output file **SQUARES.OUT** should contain the number of squares that are visible from the origin.

Examples

SQUARES.IN

```
3
2 6 3
1 4 1
3 4 1
```

SQUARES.OUT

```
3
```

SQUARES.IN

```
4
1 2 1
3 1 1
2 4 2
3 7 1
```

SQUARES.OUT

```
2
```

CARDS

Problem description

Alice and Bob have a set of N cards labelled with numbers 1 ... N (so that no two cards have the same label) and a shuffle machine. We assume that N is an odd integer.
The shuffle machine accepts the set of cards arranged in an arbitrary order and performs the following operation of **double shuffle**: for all positions i, $1 \leq i \leq N$, if the card at the position i is j and the card at the position j is k, then after the completion of the operation of double shuffle, position i will hold the card k.
Alice and Bob play a game. Alice first writes down all the numbers from 1 to N in some random order: $a_1, a_2, \ldots, a_N$. Then she arranges the cards so that the position a_i holds the card numbered a_{i+1}, for every $1 \leq i \leq N-1$, while the position a_N holds the card numbered a_1.
This way, cards are put in some order $x_1, x_2, \ldots, x_N$, where x_i is the card at the i^{th} position.
Now she sequentially performs S double shuffles using the shuffle machine described above. After that, the cards are arranged in some final order $p_1, p_2, \ldots, p_N$ which Alice reveals to Bob, together with the number S. Bob's task is to guess the order $x_1, x_2, \ldots, x_N$ in which Alice originally put the cards just before giving them to the shuffle machine. Your task is to write a computer program to do this.

Input data

The first line of the input file **CARDS.IN** contains two integers separated by a single blank character: the odd integer N, $1 \leq N \leq 1000$, the number of cards, and the integer S, $1 \leq S \leq 1000$, the number of double shuffle operations.
The following N lines describe the final order of cards after all the double shuffles have been performed such that for each i, $1 \leq i \leq N$, the $(i+1)^{st}$ line of the input file contains p_i (the card at the position i after all double shuffles).

Output data

The output file **CARDS.OUT** should contain N lines which describe the order of cards just before they were given to the shuffle machine.
For each i, $1 \leq i \leq N$, the i^{th} line of the output file should contain x_i (the card at the position i before the double shuffles).

Examples

CARDS.IN

```
5 2
4
1
5
3
2
```

CARDS.OUT

```
2
5
4
1
3
```

CARDS.IN

```
7 4
6
3
1
2
4
7
5
```

CARDS.OUT

```
4
7
5
6
1
2
3
```

Contract

Problem description

We are given a sequence of N positive integers a = [a_1, a_2, ..., a_N] on which we can perform contraction operations. One contraction operation consists of replacing adjacent elements a_i and a_{i+1} by their difference a_i-a_{i+1}. For a sequence of N integers, we can perform exactly N-1 different contraction operations, each of which results in a new (N-1) element sequence.

Precisely, let `con(a,i)` denote the (N-1) element sequence obtained from [a_1, a_2, ..., a_N] by replacing the elements a_i and a_{i+1} by a single integer a_i-a_{i+1} :

con(a,i) = [a_1, ..., a_{i-1}, a_i-a_{i+1}, a_{i+2}, ..., a_N]

Applying N-1 contractions to any given sequence of N integers obviously yields a single integer.

For example, applying contractions 2, 3, 2 and 1 in that order to the sequence [12,10,4,3,5] yields 4, since:

`con([12,10,4,3,5],2) = [12,6,3,5]`

The difference between the 2^{nd} integer and the 3^{rd} integer becomes the new 2^{nd} integer.

`con([12,6,3,5]    ,3) = [12,6,-2]`

The difference between the 3^{rd} integer and the 4^{th} integer becomes the new 3^{rd} integer.

`con([12,6,-2]     ,2) = [12,8]`

`con([12,8]        ,1) = [4]`

Given a sequence a_1, a_2, ..., a_N and a target number T, the problem is to find a sequence of N-1 contractions that applied to the original sequence yields T.

Input data

The first line of the input file **CONTRACT.IN** contains two integers separated by blank character.

The integer N, $1 \le N \le 100$, the number of integers in the original sequence, and the target integer T, $-10000 \le T \le 10000$.

The following N lines contain the starting sequence: for each i, $1 \le i \le N$, the $(i+1)^{st}$ line of the input file contains integer a_i, $1 \le a_i \le 100$.

Output data

Output file **CONTRACT.OUT** should contain N-1 lines, describing a sequence of contractions that transforms the original sequence into a single element sequence containing only number T. The i^{th} line of the output file should contain a single integer denoting the i^{th} contraction to be applied. You can assume that at least one such sequence of contractions will exist for a given input.

Examples

CONTRACT.IN

```
4 5
10
2
5
2
```

CONTRACT.OUT

```
1
2
1
```

CONTRACT.IN

```
5 4
12
10
4
3
5
```

CONTRACT.OUT

```
2
3
2
1
```