# Problem A. Agrarian Reform

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

The King of Squaredom is planning the agrarian reform. The Squaredom has the form of rectangle of $m \times n$ squares. Squares are identified by pairs $(x, y)$ where $x$ ranges from 1 to $m$, and $y$ ranges from 1 to $n$. Each square is either occupied by a peasant's house, or contains a swamp, or is a field. The King would like to assign peasants to fields, so that each peasant was assigned to exactly one field, and each field was assigned as most one peasant.

The King asked his Minister of Agronomy to prepare the list of peasants. After that he would assign them to fields. The Private Counselor of the King has found out the algorithm the King will use to assign peasants to fields.

The King would look through the peasants in order they are listed by the Minister of Agronomy. For each peasant he would find the closest to his house field that has no peasant assigned to it yet. That field would be assigned to this peasant. If there are several such fields, the field which has the smallest $x$ will be chosen, if there are still several such fields, the field which has the smallest $y$ among them will be chosen. The distance between squares $(x_1, y_1)$ and $(x_2, y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$.

The Minister of Agronomy would like to order peasants in such a way that the sum of distances between peasant and the field he is assigned to for all peasants were as small as possible. Help him to find such order.

## Input

The first line of the input file contains four integer numbers: $m$, $n$, $k$ and $s$ — the size of the field, the number of peasants, and the number of swamps, respectively ($1 \le m, n \le 20$, $1 \le k \le mn/2$, $0 \le s \le mn - 2k$). The following $k$ lines contain coordinates of squares where peasants live, the $i$-th of these lines contains two integer numbers $x_i, y_i$ ($1 \le x_i \le m$, $1 \le y_i \le n$). No two peasants live in the same square.

The following $s$ lines contain coordinates of squares containing swamps.

## Output

Output $k$ numbers — the order in which the Minister of Agronomy should order peasants so that the King assigned them to the fields in the optimal way.

## Example

| agrarian.in | agrarian.out |
|---|---|
| 3 5 5 0<br>2 3<br>2 4<br>1 3<br>2 2<br>3 3 | 3 4 2 1 5 |

# Problem B. Astronomy Problem

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

Flatland astronomy scientists are going to start permanent observations of $n$ stars. The universe where the Flatland planet is located is two-dimensional, so each star is characterized by two coordinates (measured in parsecs).

The scientists are looking for the point at space to put the observation station at. If two stars would be viewed at a small angle from the observation point, various characteristics of the stars that are to be investigated would interfere, so the scientists would like to find such point that the smallest angle between the two stars viewed from that point, were as large as possible.

Also the observation station must not be too close to any star. The distance from the station to the closest star must be at least 1 parsec.

Help the scientists to find the location for their station.

## Input

The first line of the input file contains $n$ — the number of stars ($3 \le n \le 10$). The following $n$ lines contain two integer numbers each — the coordinates of stars (coordinates do not exceed $10^3$ by their absolute values).

## Output

Output two real numbers — the coordinates of the point where scientists should put their station. If there are several solutions, output any one.

Verifying program will use precision of $10^{-5}$ when making comparisons of floating point numbers.

## Example

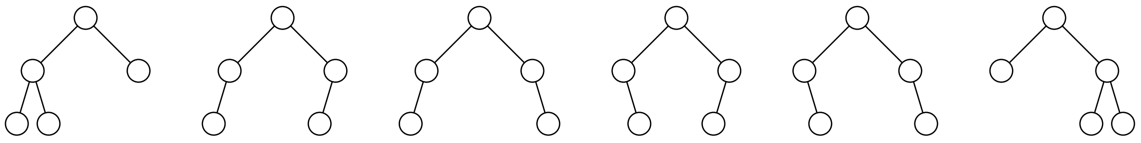
| astronomy.in | astronomy.out |
|---|---|
| 4<br>0 0<br>10 0<br>0 10<br>10 10 | 5 5 |

# Problem C. AVL Trees

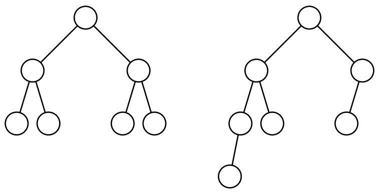
| | |
|---|---|
| Input file: | `avl.in` |
| Output file: | `avl.out` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

AVL trees invented by Russian scientists Adelson-Velskiy and Landis are used for *sorted collection* data structure. The rooted binary tree is called *balanced* if for each vertex the height of its left subtree and the height of its right subtree differ by at most one. The balanced binary search tree is called the AVL tree.

There can be several AVL trees with the given number of vertices. For example, there are 6 AVL trees with 5 vertices, they are shown on the picture below.



Also the tree with the given number of vertices can have different height, the picture below shows AVL trees with 7 vertices that have height 2 and 3, respectively.



Given $n$ and $h$ find the number of AVL trees that have $n$ vertices and height $h$. Since the answer can be quite large, return the answer modulo 786 433.

## Input

Input file contains $n$ and $h$ ($1 \le n \le 65\,535$, $0 \le h \le 15$)

## Output

Output one number — the number of AVL trees with $n$ vertices that have height $h$, modulo 786 433.

## Example

| `avl.in` | `avl.out` |
|---|---|
| `7 3` | `16` |

## Note

Note that 786 433 is prime, and $786\,433 = 3 \cdot 2^{18} + 1$.

## Problem D. Block Edit Distance

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

Recently TopCoder has run a Marathon Match 18 that was intended to support Wikipedia project. The problem that was suggested to the participants had a subproblem — to find a specially defined edit distance between the two words. In this problem we will consider a simplified version of the MM18 problem, but unlike in the Marathon Match we will require the exact solution.

You are given two words $S$ and $T$. First you are allowed to choose some non-intersecting subwords of $S$ and remove them. Each subword you remove costs you $b$. Let the resulting word be $Z$.

After that you must find a standard edit distance between $Z$ and $T$. To do it, you must find the instruction sequence that converts $Z$ to $T$. The allowed instructions are “`I`” — insert, “`D`” — delete, and “`C`” — copy.

Consider that you have two pointers, initially the first pointer is at the first character of $Z$ and the second pointer is at the first character of $T$. “`I`” instruction moves the second pointer one character to the right. “`D`” instruction moves the first pointer one character to the right. “`C`” instruction is applicable only when the two pointers are at equal characters, it moves both pointers one character to the right. Each “`I`” instruction costs $i$, each “`D`” instruction costs $d$, each “`C`” instruction costs $c$.

Find the way to transform $S$ to $T$ in the described way with the smallest cost.

### Input

The first line of the input file contains four integer numbers: $b$, $i$, $d$ and $c$ ($0 \le b, i, d, c \le 10\,000$). The second line contains $S$. The third line contains $T$. The length of each of $S$ and $T$ doesn't exceed 3 000.

### Output

The first line of the output file must contain $K$ — the cost of converting $S$ to $T$ in the described way. The second line must contain $n$ — the number of subwords of $S$ to be removed to form $Z$. The following $n$ lines must contain two integer numbers each — the inclusive ranges of subwords in $S$ to be removed. Characters are numbered from 1.

The last line must contain the sequence of instructions to convert $Z$ to $T$.

### Example

| block.in | block.out |
|---|---|
| `3 1 1 0`<br>`ABCDEFGHIJKLMN`<br>`BCDEFZZZZKLM` | `9`<br>`1`<br>`7 10`<br>`DCCCCCIIIICCCD` |

## Problem E. Cryptography

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

Professor Klever is working on cracking the new cipher *Formally Incrackable Generaly Virtual Applied Masking* (FIGVAM). After investigating the cipher he reduced the problem of restoring the encryption key to the following problem: find two numbers $x$ and $y$ ($1 \le x, y \le n$) such that:

- $x \wedge y = y$;
- $(ax + by) \oplus (ay + bx)$ is maximal.

Here $p \wedge q$ means bitwise "and" of $p$ and $q$, and $p \oplus q$ means bitwise "exclusive or" of $p$ and $q$.

You are given $n$, $a$ and $b$. Help professor to find $x$ and $y$.

### Input

Input file contains $n$, $a$ and $b$ ($1 \le n \le 100\,000$, $0 \le a, b \le 2000$).

### Output

Output $x$ and $y$ that satisfy the given conditions.

### Example

| crypto.in | crypto.out |
|---|---|
| 20 2 3 | 15 10 |

# Problem F. Independent Set

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

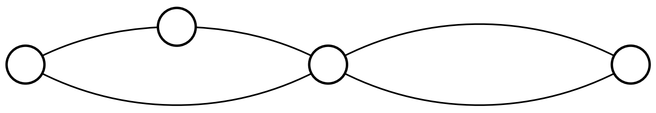
The notion of series-parallel graph is very important in electrical engineering and other applied sciences. Such graphs are used to represent electrical schemes, data networks, etc.

In this program a graphs may contain parallel edges.

The series-parallel graph is defined recursively as follows.

1. The two-vertex graph with vertices $s$ and $t$ and an edge $st$ is series-parallel graph with source $s$ and sink $t$. Let us denote such graph as $g$.
2. If $G_1$ is a series-parallel graph with source $s_1$ and sink $t_1$ and $G_2$ is a series-parallel graph with source $s_2$ and sink $t_2$, then the graph obtained by merging together vertices $t_1$ and $s_2$ is a series-parallel graph with source $s_1$ and sink $t_2$, denoted as $SG_1G_2$.
3. If $G_1$ is a series-parallel graph with source $s_1$ and sink $t_1$ and $G_2$ is a series-parallel graph with source $s_2$ and sink $t_2$, then the graph obtained by merging together vertices $s_1$ and $s_2$ (denote the new vertex as $s_{12}$), and merging together vertices $t_1$ and $t_2$ (denote the new vertex as $t_{12}$) is a series-parallel graph with source $s_{12}$ and sink $t_{12}$, denoted as $PG_1G_2$.

Using the above definition recursively, we can obtain large series-parallel graphs. For example, the graph on the figure below is denoted as $SPSgggPgg$.



You are given the series-parallel graph. Find the size of the maximal independent set in it. The independent set is the set of vertices no two vertices in which are connected by an edge.

## Input

The input file contains one line — the description of a series-parallel graph. The description is the correct expression which contains only letters '`S`', '`P`' and '`g`'. The length of the expression doesn't exceed 100 000.

## Output

Output one integer number — the size of the maximal independent set in the given graph.

## Example

| `independent.in` | `independent.out` |
|---|---|
| `SPSgggPgg` | `2` |

## Problem G. 3D Knight

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

The travelling knight problem is the famous problem that is used to show various aspects of backtracking when teaching programming. But the classic 2-dimensional version of this problem is already well studied, so let us move to 3D version of the problem.

You are given an $a \times b \times c$ parallelepiped. A chess knight can reside in any unit cube of it. The goal is to put the knight at one of the corner cubes of the parallelepiped, and find its path that visits each unit cube exactly once.

Let us denote unit cubes by three coordinates, ranging from 1 to $a$, 1 to $b$ and 1 to $c$, respectively. The knight can move from a cube to another cube if one coordinate of the destination cube is the same, one differs by exactly one, and one differs by exactly two (the standard chess knight move). The path must start at a cube $(1, 1, 1)$.

## Input

The input file contains three integer numbers $a$, $b$ and $c$ ($1 \le a, b, c \le 5$).

## Output

If the required path exists, output "`YES`" at the first line of the output file. After that list the cubes in order the are visited along the path — $abc$ lines, each of them must contain three numbers — the coordinates of the corresponding cube.

If there is no required path, output "`NO`" at the first line of the output file.

## Example

| `knight.in` | `knight.out` |
|---|---|
| `4 3 2` | `YES`<br>`1 1 1  3 2 1`<br>`1 3 1  2 1 1`<br>`4 1 2  2 2 2`<br>`4 2 1  2 3 1`<br>`3 1 1  1 2 1`<br>`3 3 1  4 1 1`<br>`4 3 2  3 1 2`<br>`1 2 2  3 3 2`<br>`2 1 2  1 3 2`<br>`3 2 2  1 1 2`<br>`2 3 2  4 3 1`<br>`2 2 1  4 2 2` |

The coordinates in the example are printed two triples on a line to save space. You should print one triple on a line.

## Problem H. Perfect Lodging

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

Every year Giggle company arranges the sponsored celebration event for the winners of the International Laughing Contest. The participants are invited to visit Giggle office in a nice Swamp Smell town.

And every year the event manager Serge faces the same problem: he must lodge $2n$ participants in $n$ twin rooms. When filing a travel request each participant indicates the list of other participants that he would agree to live in one room with.

Before making arrangements, Serge needs to divide all participants to pairs, so that each participant lives in a room with the one from the list he specified in his travel request.

Help him to find out whether it is possible.

### Input

The first line of the input file contains $2n$ — the number of participants ($2 \le 2n \le 200$). The following $2n$ lines describe participant preferences, each line starts with $k_i$ — the number of other participants, the $i$-th one would agree to live with, followed by $k_i$ integer numbers — the numbers of the corresponding participants. All participants are numbered from 1 to $n$ in order they are given in the input file.

### Output

Output "`YES`" if it is possible to lodge all participants with respect to their requests, or "`NO`" if it is not.

### Example

| perfect.in | perfect.out |
|---|---|
| 4<br>2 2 3<br>2 3 4<br>3 1 2 4<br>3 1 2 3 | YES |
| 4<br>1 2<br>1 3<br>1 4<br>1 1 | NO |

In the first example Serge can, for example, lodge participants 1 and 3 together, and 2 and 4 together.

# Problem I. Hungry Queen 2

| | |
|---|---|
| Input file: | `queen2.in` |
| Output file: | `queen2.out` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Hungry queen is standing at the cell $(0, 0)$ of an infinite chessboard. There are also $n$ pawns on the chessboard, numbered from 1 to $n$, the $i$-th pawn is located at the cell $(x_i, y_i)$.

The queen is going to take as many pawns as possible. She's going take pawns in order they are numbered: pawn 1, pawn 2, etc. All her moves must follow chess queens rules — the queen can move only horizontally, vertically and diagonally. She is not allowed to jump over pawns. The pawns don't move.

Help the queen to find the maximal number of successive pawns she can take.

## Input

The first line of the input file contains $n$ — the number of pawns $(1 \le n \le 100\,000)$. The following $n$ lines contain coordinates of pawns (coordinates do not exceed $10^9$ by their absolute values).

No pawn is located at the cell $(0, 0)$. No two pawns occupy the same cell.

## Output

Output one number — the number of successive pawns the queen can take.

## Example

| queen2.in | queen2.out |
|---|---|
| 4<br>0 2<br>1 1<br>1 -3<br>1 0 | 2 |

# Problem J. Trip Expenses

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

In Janap business trip expenses compensation is not based on the travel documents that the person presents to the accountant, but rather on the starting point of the trip and the destination of the trip. The means of transportation for the trip can be chosen by the one who is assigned to the trip on his own.

Of course, those who would like to save money choose the cheapest way of travelling between cities. However such way can often be composed of several segments and therefore take longer.

The Ministry of Finances of Janap is investigating the average number of trip segments required to get from one city to another. They suppose that the travelling person chooses the cheapest path, and among those he chooses the path with the smallest number of segments.

Help them to find that out.

## Input

The first line of the input file contains two integer numbers $n$ and $m$ ($2 \le n \le 300$, $1 \le m \le 20\,000$) — the number of cities in Janap and the number of possible ways to get from one city to another — trip segments. The following $m$ lines describe trip segments, each segment is described by three integer numbers $a$, $b$ and $c$ — the cities it connects and the cost of travelling along this segment. Each segment can be traveled in either direction. There can be several segments between two cities. The cost of travelling is positive and doesn't exceed $10^9$. It is possible to get from any city to any other.

## Output

Output one floating point number — the average number of segments required to get from one city to another by cheapest ways. You answer must be accurate up to $10^{-5}$.

## Example

| trip.in | trip.out |
|---|---|
| 3 3<br>1 2 10<br>2 3 2<br>1 3 3 | 1.33333333333333333 |

In the example the cheapest path from 1 to 2 has two segments ($1 \to 3 \to 2$), the cheapest paths from 1 to 3 and from 2 to 3 have one segment each. The average number of segments if $(2 + 1 + 1)/3 = 4/3$.