SJAARS KAMPIOENSCHAP PROGRAMMEREN 2015

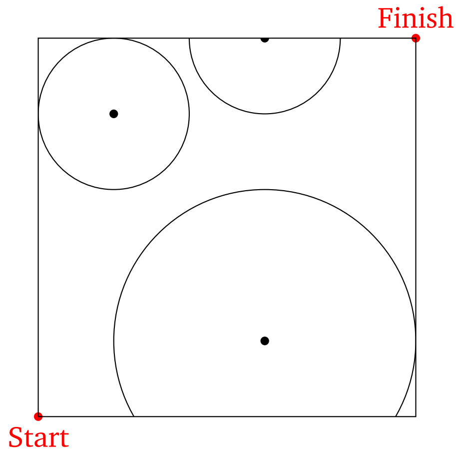
CONTEST PROBLEM SET

MARCH 26, 2015

A - Alcatraz

After breaking into the database of the TU Delft and releasing sensitive information to the public, Anton the cyber criminal got the CIA looking for him. After being arrested, Anton was transported to the heavily guarded prison Alcatraz. Disabling the security system was a piece of cake, so he could easily get outside of the prison walls. However, behind these walls a number of guards are positioned and they will immediately sound the alarm if they spot any escaped prisoner.

To escape, Anton needs to cross a field with width w and height h that is being guarded by n guards, without being spotted by any of the guards. He starts at position $(0, 0)$ and can only escape if he reaches position (w, h) without being spotted. Even if the criminal is only spotted at position (n, n) the guards will sound the alarm and he will be captured. Some guards have better views that others, so every guard i has an associated range of view r_i. If the distance between the guard and the escaped prisoner is r_i or less, the guard will sound the alarm and the prisoner will return to his cell.
Your job is now to find out whether Anton can escape Alcatraz or if he cannot escape without being spotted. If he can escape, output ESCAPE. Otherwise output NO ESCAPE.



Input

- A line with 3 integers:
  - w: the width of the area ($1 \leq w \leq 100$).
  - h: the height of the area ($1 \leq h \leq 100$).
  - g: the number of guards ($0 \leq g \leq 100$).
- n lines ($i = 1 \ldots n$) with 3 integers each:
  - x_i: the x-coordinate of the i^{th} guard ($0 \leq x_i \leq w$).
  - y_i: the y-coordinate of the i^{th} guard ($0 \leq y_i \leq h$).
  - r_i: the range of the i^{th} guard's vision ($0 \leq r_i \leq 100$).

Output

ESCAPE if the prisoner can escape Alcatraz, NO ESCAPE otherwise.

Examples

input 1	output 1
5 5 3 3 1 2 3 5 1 1 4 1	ESCAPE

input 2	output 2
5 5 3 3 1 2 3 4 1 1 4 1	NO ESCAPE

This page is intentionally left (almost) blank.

B - Bad English

The Dutch are known to be bad at English sometimes, especially when it comes to pronunciations and translations. A classic example is the football manager Louis van Gaal. During a press conference he said “That is another cook.”, which is a literal translation of the dutch figure of speech ”Dat is andere koek.”. These so called “stonecoal English” sentences are sentences that are literally translated from Dutch to English.

Since you are responsible for the next broadcast of van Gaal, you wish to prevent the occurrence of these “stonecoal English” sentences. Luckily, this broadcast is not live and hence when you detect a faulty sentence, you can simply redo that part. During the recording a program automatically records all sentences van Gaal says. Your task is to write a program that, given a dictionary of words, the output of the recording by the program and the Dutch equivalent of the recording, checks if sentences are said in “stonecoal English”. As said before, a “stonecoal English” sentence is a sentence that is literally translated. If different, additional or fewer words are used or if the sentence contains words that are spelled incorrectly, then we may assume that an attempt was made by van Gaal to speak correct English and thus we won’t mark them as “stonecoal English”. You do not have to take (correct) punctuation into account.

Input

Four strings r, t, d and e which contain between 1 and 10^4 (inclusive) words separated by spaces. The first string r is the recording of van Gaal. The second string t is the Dutch equivalent of what van Gaal meant. The third string d lists the Dutch words said in t without repetition, in lowercase and are sorted alphabetically. The fourth string e contains the English translation of the words in d in the same order as they occur in d (this means that e and d contain an equal amount of words), again in lowercase.

Output

Output ”STONECOAL” if the sentence was literally translated and hence belong to the domain of “stonecoal English”. Output “VALID” otherwise.

Examples

input 1	output 1
Make that the cat wise. Maak dat de kat wijs. dat de kat maak wijs that the cat make wise	STONECOAL

input 2	output 2
He kicked the ball very hard. Hij schopte hard tegen de bal. bal de hard hij schopte tegen ball the hard he kicked against	VALID

input 3	output 3
That hits like a tang on a pig. Dat slaat als een tang op een varken. als dat een op slaat tang varken if that a on hit pliers pig	VALID

C - Composius' Wrath

After many years of praising their god Primos, the very wealthy land of Primozia had a large number of cities that were connected by many roads, enabling traders to travel from every city to every other city in very little time. This caused the god Composius to be very displeased, because he does not like to see such wealth in a country that idolizes prime numbers. Therefore he cast his wrath upon the land of Primozia and destructed all their trade routes by the force of an incredible tsunami.

After hearing about this, the first priority of the pharaoh of Primozia was to rebuild the network of roads such that all trading can continue. However, in order to connect every city with every other city as soon as possible, the pharaoh orders to rebuild the road network with as few roads as possible, such that a path exists from every city to every other city. Additionally, he wants to rebuild as many roads with prime lengths as possible, because prime numbers are sacred in the land of Primozia.

Input

On the first line two integers, the number of cities c ($1 \leq c \leq 100$) and the number of roads r ($c - 1 \leq r \leq 10^4$). On the next r lines, three integers a, b and w are given, with a and b the endpoints of a road and w ($1 \leq w \leq 10^4$) the length of that road.

Output

One line with two integers p and q. With p the number of roads with prime length and q the number of roads with non-prime length.

Examples

input 1	output 1
7 9 1 2 1 2 3 7 2 4 5 2 6 5 3 4 9 4 5 6 4 6 11 5 7 8 5 6 13	4 2

input 2	output 2
7 12 1 2 5 1 6 64 1 7 12 2 3 11 2 7 21 3 4 17 3 7 32 4 5 83 4 7 18 5 6 113 5 7 19 6 7 7	6 0

D - Deal or No Deal

In the popular television show *Deal or No Deal*, produced by Endemol, participants have a chance of winning high amounts of money. The game is derived from the Dutch game *Miljoenenjacht*. It is often played with 26 boxes containing certain sums of money. First, the player selects a random box which he keeps. The contestant then begins choosing boxes that are to be removed from play. The amount inside each chosen box is immediately revealed after the contestant has chosen a box. The amount revealed cannot be inside the case the contestant has initially chosen. Throughout the game, after a predetermined number of cases have been opened, the banker offers the contestant an amount of money to quit the game. The bank has the following simple formula for making a offer: they calculate the average of all amounts of money in the unopened boxes (excluding the chosen box by the contestant) and multiply this average by 0.9. This offer is presented to the player and he then answers the famous question, choosing:

- *deal*, accepting the offer presented and ending the game. If the contestant accepts the offer, he or she cannot accept subsequent offers the bank makes later in the game.
- *no deal*, rejecting the offer and continuing the game.

This process of removing cases and receiving offers continues, until either the player deals, or all offers have been rejected and the values of all unselected boxes are revealed. Should a player end the game by taking a deal, the game is continued from that point to see how much the player could have won by continuing the game. Depending on subsequent choices and offers, it is determined whether or not the contestant made a good deal, i.e. won more than if the game were allowed to continue.

The game producer keeps good track of the amount of money that has been paid to contestants and the amount of money contestants could have won by playing to the end of the game by refusing every offer the bank makes. Recently, this database got corrupted. Luckily, they found some documents in a hidden archive, describing the progress of each of their games and specifying the choices contestants make. Each documents gives information about a single game. Using this document, it is possible to reconstruct the amount of money that the contestant actually won and could have won by playing to the very end of the game. They hired you to write a program that computes these two amounts of money.

Input

- On the first line there is one integer n: the number of boxes in the game ($1 \leq n \leq 1000$).
- After that, there are n space-separated numbers, each describing the amount of money in box i ($0.01 \leq m[i] \leq 5000000$ where $m[i]$ is the amount of money in box i).
- After that, there are two space-seperated integers a and q, a being the box that is chosen by the contestant (the first box has index 1) and q being the number of actions that have been executed during the game.

- After that, q lines follows with each line specifying a possible action during the game. Each action is specified by a single uppercase character. The several actions are:
  - The contestant opens a box, specified by the character *O*. The character is followed by the index of the box that is opened (the first box has index 1).
  - The bank makes an offer. This is indicated by the character *B*, followed by the choice of the contestant (deal or no deal). A *D* follows if the contestant accepts the offer and a *N* follows if the contestant refuses the offer.

Output

Two lines with on the first line, the amount of money the contestant could have won by playing to the end of the game. On the second line, output the amount of money that the contestant has actually won. The numbers should be rounded to two decimals.

Examples

input 1	**output 1**
7 1.00 2.00 3.00 4.00 5.00 6.00 7.00 4 8 O 3 O 2 O 1 B N O 7 O 5 B D O 6	4.00 5.40

input 2	**output 2**
7 100.00 500.00 1000.00 2000.00 10000.00 1000000.00 5000000.00 7 8 O 3 O 2 O 1 B N O 7 O 5 B N O 6	5000000.00 5000000.00

E - Excellent Grades

In order to receive his propedeutic degree *cum laude*, the weighted average of all of Eddie's grades must be at least 8.0. Of course he must also have passed all of his courses, which requires a grade of at least 5.8. With only one exam to go, Eddie is wondering if he still has a shot at achieving *cum laude*.

Given all of the grades for Eddie's previous courses and their weights, and the weight of the final exam, check if Eddie can still obtain his degree *cum laude*. If he can, output the grade Eddie requires for his final exam to achieve this. Otherwise output IMPOSSIBLE.

Input

- A line with 1 integer w_u ($1 \leq w_u \leq 30$), the weight of the final exam
- A line with 1 integer n ($1 \leq n \leq 100$), the number of other grades
- n lines with 1 double (1 decimal) g_i ($1 \leq g_i \leq 10$) and 1 integer w_i ($1 \leq w_i \leq 30$) where g is the grade of exam i and w_i is the weight of exam i.

Output

If Eddie can achieve *cum laude*, print the grade g ($1 \leq g \leq 10$) he has to achieve on his final exam, rounded to 1 decimal. Otherwise print IMPOSSIBLE.

Examples

input 1	output 1
3 3 8.0 2 7.0 1 7.0 4	9.7

input 2	output 2
2 4 8.0 2 6.1 1 8.3 1 7.1 3	IMPOSSIBLE

F - Floor Tiling

Your friend John has recently moved to the city of Delft. During the decoration of one of the rooms in his house, he decided that he does not like the current floor in the room. He has seen several interesting floor patterns but there's one pattern that has his favor: a floor that is covered domino planks. A domino plank is a special type of plank that has a width of one meter and a height of two meters. The planks can be laid out horizontally or vertically and cannot be sawed into smaller pieces. Now John still has to decided how the floor is going to look like because there are many possibilities to cover the whole floor. For example, if the room of John is 3x8 meters, there are already 153 possible ways of covering the floor!

John would like to know how many possibilities there are for each room in his house. Since he's very smart, he has come up with a program that outputs, given the height and width of the room, the number of ways to cover the floor of the room with domino tiles. He challenges you to do the same, but he made the problem easier: you can assume that the width of the rooms is always 3 meter. Moreover, you can assume that the width of the room is even (otherwise the floor cannot be fully covered with domino planks). Can you solve the challenge of John and tell how many possible coverings there are for a 3 x $2n$ floor?

Input

A single integer n, $1 \leq n \leq 100$.

Output

The number of different ways in which a 3 x $2n$ floor can be tiled with domino planks.

Examples

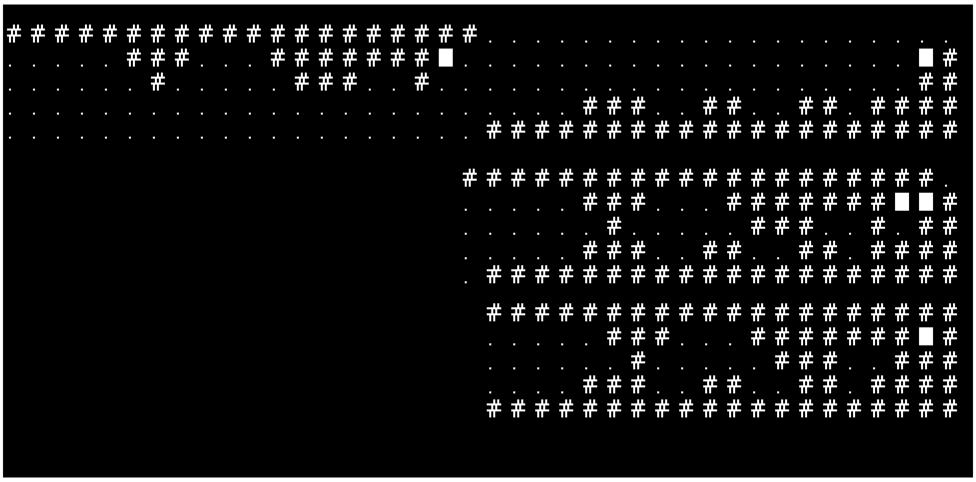
input 1	output 1
`1`	`3`

input 2	output 2
`15`	`299303201`

G - Grand Opening

Working at a lock recycling company can be a pain, especially when the locks need to be unlocked and reusable and people forgot to turn in their keys. A friend that works at such a company has found a solution to this problem: a bump key. Your friend has a list of specifications for the tumblers in each lock, and needs your help. You are tasked with finding out which locks he should attempt to open using a bump key, and which locks are for the furnace.

The key and locks are made of the character "#", and the rest is indicated by a dot. In the example below, you can see that the key on the left doesn't fit the lock on the right. The parts where the key doesn't fit are marked with a "■".



Input

- A line with three space-separated integers n w and h ($1 \leq n \leq 1000, 1 \leq w, h \leq 100$), indicating the number of locks, and the width and height of the key and locks.
- n locks, where each lock consists of h lines of w characters "#" or ".".
- h lines of w character "#" or ".", representing the key.

Output

A single integer, the number of locks the given key fits in.

Examples

input 1

```
1 20 5
....................
....................
....................
####################
....................
....................
....................
....................
####################
....................
```

output 1

```
0
```

input 2

```
3 20 5
....................
...................#
..................##
....###..##..##.####
####################
....................
...................#
..........#........#
....###..##..##.#..#
####################
....................
...................#
..##...............#
..#####..##..##.#..#
####################
####################
.....###...########.
......#.....###..#..
....................
....................
```

output 2

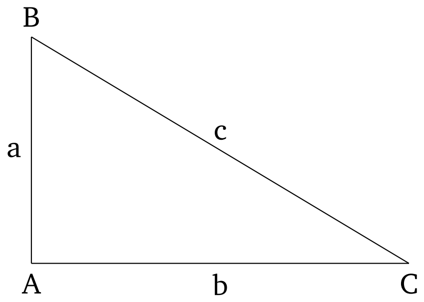
```
1
```

H - Hypotenuse

Martijn is a math teacher and next weeks class is about geometry. He would like to have some tools for his pupils to practice the course material. One of his ideas is to have a program that can solve calculate the unknown length of one side of a right angled triangle, given the other two sides. Pythagorean theorem states that, in a right angled triangle, the square of the hypotenuse is equal to sum of the squares of the other two sides. Your task is to make a solver that uses Pythagorean theorem.

$$\mathrm{AB}^2 + \mathrm{AC}^2 = \mathrm{BC}^2$$

$$a^2 + b^2 = c^2$$



Input

The input consists out of two integers ($0 < x, y \leq 2.3 \cdot 10^4$) and a question mark ('?'), in any order. The order of the three parts of the input match with a, b and c.

Output

The solution to '?' in its most simple form and in LaTeX notation. For example: $\sqrt{20}$ needs to be simplified to $2 \cdot \sqrt{5}$. The formula $2 \cdot \sqrt{5}$ is in LaTeX written as "`2 \cdot \sqrt{5}`".

Examples

input 1	output 1
`6 5 ?`	`\sqrt{61}`

input 2	output 2
`? 4 5`	`3`

input 3	output 3
`3 9 ?`	`3 \cdot \sqrt{10}`