FIU 2023 HSPC FIU 2023 High School Programming Competition

Florida International University
High School Programming Competition

Novice Division 2

Problems

A. Army Strength

The next MechaGodzilla invasion is on its way to Earth. And once again, Earth will be the battleground for an epic war. MechaGodzilla's army consists of many nasty alien monsters, such as Space Godzilla, King Gidorah, and MechaGodzilla herself. To stop them and defend Earth, Godzilla and her friends are preparing for the battle.

Each army consists of many different monsters. Each monster has a strength that can be described by a positive integer. The larger the value, the stronger the monster. The war will consist of a series of battles. In each battle, the weakest of all the monsters that are still alive is killed. If there are several weakest monsters, but all of them in the same army, one of them is killed at random. If both armies have at least one of the weakest monsters, a random weakest monster of MechaGodzilla's army is killed. The war is over if in one of the armies all monsters are dead. The dead army lost, the other one won. You are given the strengths of all the monsters. Find out who wins the war.

Input

The first line of the input file contains an integer $T \leq 50$ specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with line containing two positive integers N_G and N_M – the number of monsters in Godzilla's and in MechaGodzilla's army. You can assume that $N_G, N_M \leq 500$.

Two lines follow. The first one contains N_G positive integers – the strengths of the monsters in Godzilla's army. Similarly, the second one contains N_M positive integers – the strengths of the monsters in MechaGodzilla's army. All strengths are at most 10000.

Output

For each test case, output a single line with a string that describes the outcome of the battle. If it is sure that Godzilla's army wins, output the string "Godzilla". If it is sure that MechaGodzilla's army wins, output the string "MechaGodzilla". Otherwise, output the string "uncertain".

Sample Input 1	Sample Output 1

`2`
 	
`1 1`
`1`
`1`
 	
`3 2`
`1 3 2`
`5 5`
	`Godzilla`
	`MechaGodzilla`

B. Bishops

Yesterday was Sam's birthday. The most interesting gift was definitely the chessboard. Sam quickly learned the rules of chess and defeated his father, all his friends, his little sister, and now no one wants to play with him any more.

So he decided to play with another birthday gift – a Book of Math Problems for Young Mathematicians. He opened the book somewhere in the middle and read the following problem: "How many knights can be placed on a chessboard without threatening each other?" After a while he realized that this was trivial and moved on to the next problem: "How many bishops can be placed on a chessboard without threatening each other?". Sam is in trouble here. He is not able to solve this problem and needs your help.

Task

Sam's chessboard has size N × N. A bishop can move to any distance in any of the four diagonal directions. A bishop threatens another bishop if it can move to the other bishop's position. Your task is to compute the maximum number of bishops that can be placed on a chessboard in such a way that no two bishops threaten each other.

Input

The input file consists of several lines. The line number i contains a positive integer representing the size of the *i-th* chessboard. The size of each chessboard is at most 1 000 000.

Output

The output file should contain the same number of lines as the input file. The *i-th* line should contain one number – the maximum number of bishops that can be placed on *i-th* chessboard without threatening each other.

Sample Input 1	Sample Output 1
2 3	2 4

C. Even Up Solitaire

The Even Up Solitaire can be played with a stack of cards each having a numerical value from 1 to 100. The cards are laid out from left to right in a row. At every step, the player is allowed to remove two adjacent cards if the sum of their values is even. The gap is then "closed" by shifting the cards to the right of the gap. The order of the remaining cards is not changed. The game stops when all cards are removed or when no more cards can be removed. The player wins when all cards are removed. If this is not possible, the player should try to minimize the number of cards remaining.

You are given the initial list of cards, in left-to-right order. Determine the minimum number of cards that remain if the player moves optimally.

Input

The input consists of one case. The first line contains an integer *n* *(1 ≤ n ≤ 100000)* giving the number of cards to follow. The second line contains *n* integers indicating the card values from left to right. Each card value is in the range 1 to 100.

Output

Print the minimum number of cards that remain if the player moves optimally.

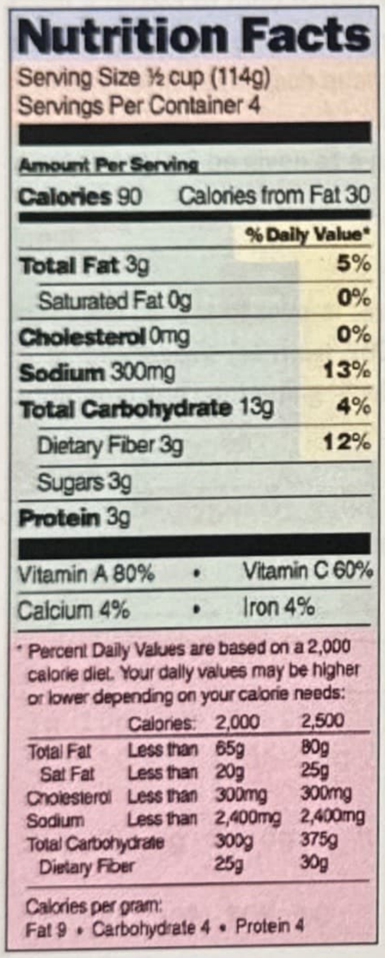
Sample Input 1	Sample Output 1

`10`
`1 2 3 4 5 6 7 8 9 10`
	`10`

Sample Input 2	Sample Output 2

`10`
`1 3 3 4 2 4 1 3 7 1`
	`2`

D. Calories from Fat



Nutrition Facts

Serving Size ½ cup (114g)
Servings Per Container 4

Amount Per Serving

Calories 90 Calories from Fat 30

	% Daily Value*
Total Fat 3g	5%
Saturated Fat 0g	0%
Cholesterol 0mg	0%
Sodium 300mg	13%
Total Carbohydrate 13g	4%
Dietary Fiber 3g	12%
Sugars 3g	
Protein 3g	

Vitamin A 80% • Vitamin C 60%
Calcium 4% • Iron 4%

* Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs:

	Calories:	2,000	2,500
Total Fat	Less than	65g	80g
Sat Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Total Carbohydrate		300g	375g
Dietary Fiber		25g	30g

Calories per gram:
Fat 9 • Carbohydrate 4 • Protein 4

Fat contains about 9 Calories/g of food energy. Protein, sugar, and starch contain about 4 Calories/g, while alcohol contains about 7 Calories/g. Although many people consume more than 50% of their total Calories as fat, most dieticians recommend that this proportion should be 30% or less. For example, in the *Nutrition Facts* label in the picture, we see that 3g of fat is 5% of the recommended daily intake based on a 2,000 calorie diet. A quick calculation reveals that the recommended daily intake of fat is therefore 60g; that is, 540 Calories or 27% Calories from fat.

Others recommend radically different amounts of fat. Dean Ornish, for example, suggests that less than 10% of total caloric intake should be fat. On the other hand, Robert Atkins recommends the elimination of all carbohydrate with no restriction on fat. It has been estimated that the average Atkins dieter consumes 61% of Calories from fat.

From a record of food eaten in one day, you are to compute the percent Calories from fat. The record consists of one line of input per food item, giving the quantity of fat, protein, sugar, starch and alcohol in each. Each quantity is an integer followed by a unit, which will be one of: g (grams), C (Calories), or % (percent Calories). Percentages will be between 0 and 99. At least one of the ingredients will be given as a non-zero quantity of grams or Calories (not percent Calories).

Input

Input will consist of several test cases (at most 150). Each test case will have one or more lines as described above (at most 100). Each test case will be terminated by a line containing '-'. An additional line containing '-' will follow the last test case.

Output

For each test case, print percent Calories from fat, rounded to the nearest integer.

Sample Input 1	Sample Output 1

`3g 10g 10% 0g 0g`
`55% 100C 0% 0g 30g`
`-`
`25g 0g 0g 0g 0g`
`-`
`1g 15% 20% 30% 1C`
`-`
`-`
	`53%`
	`100%`
	`32%`

E. Happy Happy Prime Prime

RILEY VASHTEE: [*reading from display*] Find the next number in the sequence:

313 331 367 ...? What?

THE DOCTOR: 379.

MARTHA JONES: What?

THE DOCTOR: It's a sequence of happy primes – **379**.

MARTHA JONES: Happy *what?*

THE DOCTOR: Any number that reduces to one when you take the sum of the square of its digits and continue iterating it until it yields 1 is a happy number. Any number that doesn't, isn't. A *happy* prime is both happy and prime.

THE DOCTOR: I dunno, talk about *dumbing down*. Don't they teach recreational mathematics anymore?

Excerpted from *"Dr. Who,"* Episode 42 (2007).

The number 7 is certainly prime. But is it happy?

$$7 \rightarrow 7^2=49$$
$$49 \rightarrow 4^2+9^2=97 \quad 97 \rightarrow 9^2+7^2=130$$
$$130 \rightarrow 1^2+3^2=10$$
$$10 \rightarrow 1^2+0^2=1$$

It is happy :-) As it happens, 7 is the smallest happy prime. Please note that for the purposes of this problem, 1 is *not* prime.

For this problem you will write a program to determine if a number is a happy prime.

Input

The first line of input contains a single integer *P, (1≤P≤10000),* which is the number of data sets that follow. Each data set should be processed identically and independently.
Each data set consists of a single line of input. It contains the data set number, *K*, followed by the happy prime candidate, *m, (1 ≤ m ≤ 10000).*

Output

For each data set there is a single line of output. The single output line consists of the data set number, *K*, followed by a single space followed by the candidate, *m*, followed by a single space, followed by YES or NO, indicating whether *m* is a happy prime.

Sample Input 1	Sample Output 1

4
1 1
2 7
3 383
4 1000
	1 1 NO
	2 7 YES
	3 383 YES
	4 1000 NO

F. Awkward Party

Martin has invited everyone he knows to celebrate his 535th birthday, and a whopping 535th people from all over the world have accepted the invitation.

When deciding the seating arrangement, Martin's mother Margarethe have decided that all the guests should be seated with *maximum awkwardness*; this is to ensure that nobody has anything meaningful to discuss during dinner, and everyone would instead silently enjoy her rather tasty coriander soup (as the saying goes; "when the food is good, conversation dies").

Margarethe knows that awkwardness is maximized if the guests are seated in a long row along a single table, in such a way that nobody sits next to someone speaking the same language as themselves. Better yet, she has defined the *awkwardness level* of a seating arrangement to be the minimum number of seats separating any two guests speaking the same language. If no two people speak the same language, the awkwardness level is defined to be *n* (the number of guests). Two seats next to each other are said to be separated by 1.

Given the languages spoken in a proposed seating arrangement, can you help Margarethe determine the awkwardness level?

Input

The first line contains an integer *n* *(1 ≤n ≤ 100000)* denoting the number of guests. On the second line follows n integers, the *i'th* of which x_i ($0 \le x_i \le 10^9$) indicating the language spoken by the guest seated at position i in the proposed arrangement (each guest speaks exactly one language).

Output

A single integer, the awkwardness level of the proposed seating arrangement.

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

Sample Input 2	Sample Output 2
3 1 2 3	3

G. Watering the Fields

Due to a lack of rain, Farmer John wants to build an irrigation system to send water between his N fields ($1 \le N \le 2000$).

Each field i is described by a distinct point (x_i, y_i) in the 2D plane, with $0 \le x_i, y_i \le 1000$. The cost of building a water pipe between two fields i and j is equal to the squared Euclidean distance between them:

$(x_i - x_j)^2 + (y_i - y_j)^2$

FJ would like to build a minimum-cost system of pipes so that all of his fields are linked together -- so that water in any field can follow a sequence of pipes to reach any other field.

Unfortunately, the contractor who is helping FJ install his irrigation system refuses to install any pipe unless its cost (squared Euclidean length) is at least C ($1 \le C \le 1,000,000$).

Please help FJ compute the minimum amount he will need pay to connect all his fields with a network of pipes.

Input
Line 1: The integers N and C.
Lines 2..1+N: Line i+1 contains the integers x_i and y_i.

Output
Line 1: The minimum cost of a network of pipes connecting the fields, or -1 if no such network can be built.

Sample Input 1	Sample Output 1

`3 11`
`0 2`
`5 0`
`4 3`
	`46`

Input Details:

There are 3 fields, at locations (0,2), (5,0), and (4,3). The contractor will only install pipes of cost at least 11.

Output Details:

FJ cannot build a pipe between the fields at (4,3) and (5,0), since its cost would be only 10. He therefore builds a pipe between (0,2) and (5,0) at cost 29, and a pipe between (0,2) and (4,3) at cost 17.

H. Rings

Dee Siduous is a botanist who specializes in trees. A lot of her research has to do with the formation of tree rings, and what they say about the growing conditions over the tree's lifetime. She has a certain theory and wants to run some simulations to see if it holds up to the evidence gathered in the field.

One thing that needs to be done is to determine the expected number of rings given the outline of a tree. Dee has decided to model a cross section of a tree on a two dimensional grid, with the interior of the tree represented by a closed polygon of grid squares. Given this set of squares, she assigns rings from the outer parts of the tree to the inner as follows: calling the non-tree grid squares "ring 0", each ring *n* is made up of all those grid squares that have at least one ring *(n−1)* square as a neighbor (where neighboring squares are those that share an edge). An example of this is shown in the figure below.

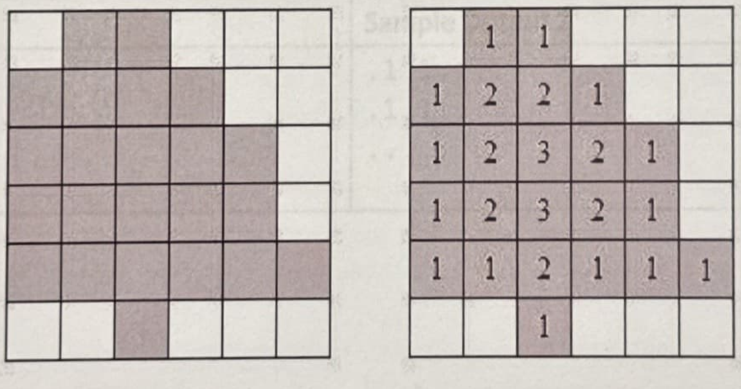


Figure D.1

Most of Dee's models have been drawn on graph paper, and she has come to you to write a program to do this automatically for her. This way she'll use less paper and save some ... well, you know.

Input

The input will start with a line containing two positive integers n m specifying the number of rows and columns in the tree grid, where *n,m≤100*. After this will be *n* rows containing *m* characters each. These characters will be either 'T' indicating a tree grid square, or '.'.

Output

Output a grid with the ring numbers. If the number of rings is less than 10, use two characters for each grid square; otherwise use three characters for each grid square. Right justify all ring numbers in the grid squares, and use '.' to fill in the remaining characters. If a row or column does not contain a ring number it should still be output, filled entirely with '.'s.

Sample Input 1	Sample Output 1
`6 6`	`...1.1......`
`.TT...`	`.1.2.2.1....`
`TTTT..`	`.1.2.3.2.1..`
`TTTTT.`	`.1.2.3.2.1..`
`TTTTT.`	`.1.1.2.1.1.1`
`TTTTTT`	`.....1......`
`..T...`	

Sample Input 2	Sample Output 2
`3 4`	`.1.1....`
`TT..`	`.1.1....`
`TT..`	`........`
`....`	

I. Pawn Shop

You run a tight ship at the pawn shop. You arrange certain items in the window to be displayed to the street. You sometimes display the same type of item multiple times. For simplicity, we think of the items on display as a sequence of values where the value represents the type of item being shown.

For example, your display could be this sequence:

1 2 6 2 7 9 8 5

After coming back from your latest vacation, you find that your staff has completely rearranged the display by moving items around. Yikes! For example, the display above could be rearranged to:

2 6 1 2 9 7 5 8

You fear this could be the cause of confusion and may scare off repeat customers. But you don't have time to move items back to their original positions.

As a compromise, you will put dividers up in the window to partition the displayed items into groups of consecutive items. Each group should be a rearrangement of the types of items that were in those positions in your preferred arrangement.

More precisely, let $a_1,\ldots,a_N$ denote the first sequence and $b_1,\ldots,b_N$ denote the second sequence. You may place dividers around $i,i+1,\ldots,j$ if $b_i,b_{i+1},\ldots,b_j$ is a rearrangement of $a_i,a_{i+1},\ldots,a_j$. You do not need to put a divider at the beginning or end of the sequence. Note, if $a_i=b_i$ then a group may be formed using just this single index i.

With the sequences above, you could place dividers (shown here as #) at three positions as indicated here:

2 6 1 # 2 # 7 9 # 5 8

It is not possible to divide the sequence into more than four groups that have this property.

Given the two sequences, determine how to partition the new sequence into the maximum possible number of groups.

Input

The first line of input contains a single integer N ($1 \le N \le 300000$), which is the length of the two sequences.
The next line contains N integers $a_1,\ldots,a_N$ ($1 \le a_i \le 10^9$), which is the original sequence.

The next line contains N integers $b_1,\ldots,b_N$ ($1 \le b_i \le 10^9$), which is the rearranged sequence. The values $b_1,\ldots,b_N$ are a rearrangement of the values $a_1,\ldots,a_N$.

Output

Display the rearranged sequence with a valid and maximum placing of dividers (#).

If there are multiple possible solutions, display any of them.

Sample Input 1	Sample Output 1

`8`
`1 2 6 2 7 9 8 5`
`2 6 1 2 9 7 5 8`
	`2 6 1 # 2 # 9 7 # 5 8`

Sample Input 2	Sample Output 2

`4`
`1 1 2 1`
`2 1 1 1`
	`2 1 1 # 1`

Sample Input 3	Sample Output 3

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

J. Forced Choice

Puff the Magic Dinosaur is a renowned magician. Due to the pandemic, he can only perform for his princess using the latest video conferencing tools. As a result, many of his usual in-person routines cannot be performed, and he must come up with a new trick.

Puff has decided to perform the following mentalist trick for the princess. First, Puff lays out N cards on the table. Each of these cards have a unique label *1,...,N.* Before the start of the show, Puff wrote down a prediction which is the label of one of the cards, put it into a sealed envelope, and mailed it to the princess.

During the show, Puff asks the princess to choose some of the remaining cards. Puff is careful to remind the princess to choose at least one but not all of the remaining cards. After the princess tells Puff which cards she chose, Puff would then say either "you chose to keep those cards" or "you chose to remove those cards". In the first case, the chosen cards are kept and all other cards are removed from the table. In the second case, the chosen cards are removed and the remaining cards are kept. This is repeated until there is only one card left. At this point Puff asks the princess to open the envelope, and magically the prediction matches the remaining card on the table.

Help Puff determine the appropriate response at each step.

Input

The first line of input contains three integers *N* $(2 \leq N \leq 200)$, which is the number of cards, *P* $(1 \leq P \leq N)$, which is the secret prediction, and *S* $(1 \leq S \leq N-1)$, which is the number of steps.

The next *S* lines describe the choices of the princess at each step. Each of these lines starts with the integer *m* $(1 \leq m \leq N-1)$, which is the number of cards chosen, followed by *m* distinct integers indicating the labels of the cards chosen. It is guaranteed that if Puff has carried out the previous steps correctly, the princess will only choose cards still on the table. Furthermore, the princess will always choose at least one but not all of the remaining cards.

It is guaranteed that there will be one card left after *S* steps if Puff has carried out all S steps correctly.

Output

For each step, display on a line either KEEP or REMOVE if Puff should keep the chosen cards or remove the chosen cards, respectively.

Sample Input 1	Sample Output 1

10 3 4
2 1 5
5 2 3 7 8 10
3 2 7 10
1 8
	REMOVE
	KEEP
	REMOVE
	REMOVE

K. Kayaking Trip

You are leading a kayaking trip with a mixed group of participants in the Stockholm archipelago, but as you are about to begin your final stretch back to the mainland you notice a storm on the horizon. You had better paddle as fast as you can to make sure you do not get trapped on one of the islands. Of course, you cannot leave anyone behind, so your speed will be determined by the slowest kayak. Time to start thinking; How should you distribute the participants among the kayaks to maximize your chance of reaching the mainland safely?

The kayaks are of different types and have different amounts of packing, so some are more easily paddled than others. This is captured by a speed factor c that you have already figured out for each kayak. The final speed v of a kayak, however, is also determined by the strengths s1 and s2 of the two people in the kayak, by the relation v=c(s1+s2). In your group you have some beginners with a kayaking strength of s_b, a number of normal participants with strength s_n and some quite experienced strong kayakers with strength s_e.

Input

The first line of input contains three non-negative integers b, n, and e, denoting the number of beginners, normal participants, and experienced kayakers, respectively. The total number of participants, b+n+e, will be even, at least 2, and no more than 100000. This is followed by a line with three integers s_b, s_n, and s_e, giving the strengths of the corresponding participants ($1 \le s_b < s_n < s_e \le 1000$). The third and final line contains $m = \frac{(b+n+e)}{2}$ integers $c_1, \ldots, c_m$ ($1 \le c_i \le 100000$ for each i), each giving the speed factor of one kayak.

Output

Output a single integer, the maximum speed that the slowest kayak can get by distributing the participants two in each kayak.

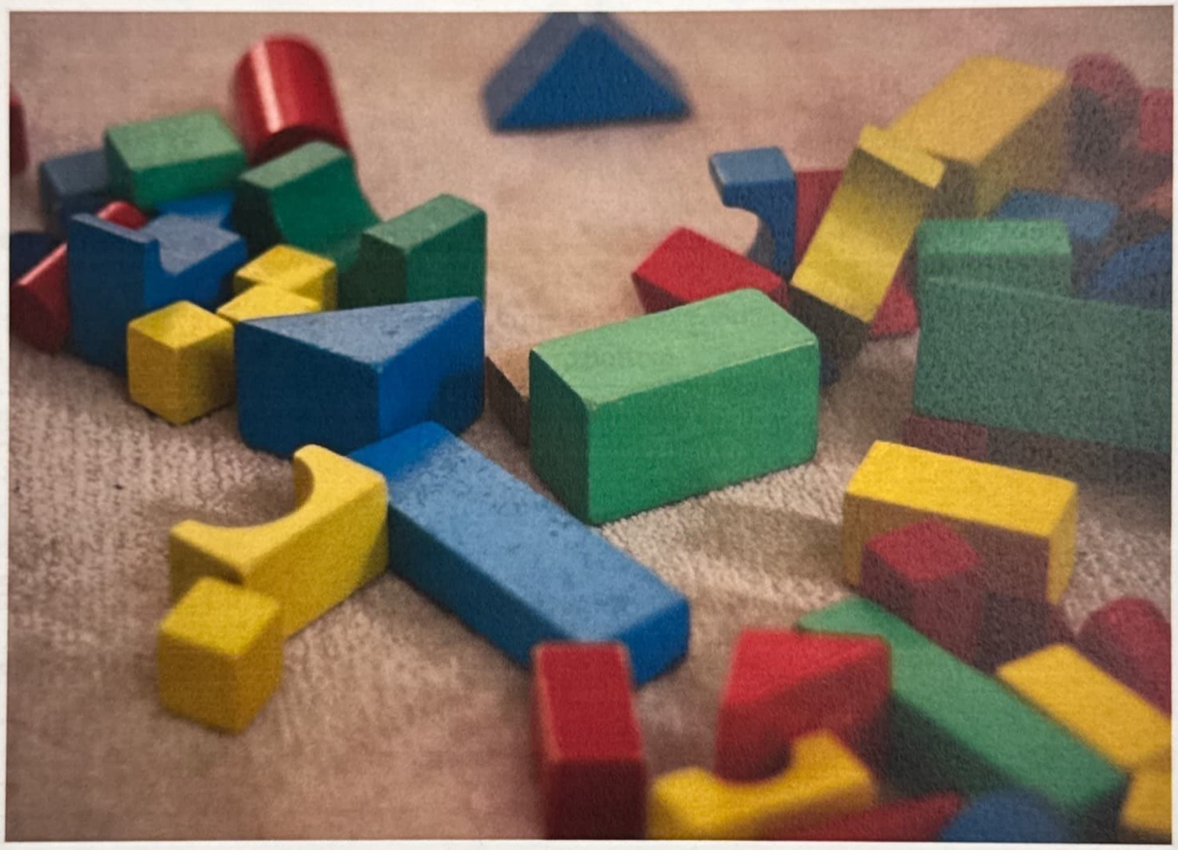
Sample Input 1	Sample output 1

3 1 0
40 60 90
18 20
	1600

Sample Input 2	Sample Output 2

`7 0 7`
`5 10 500`
`1 1 1 1 1 1 1`
	`505`

L. Adolescent Architecture



Building blocks by Thaliesin on Pixabay

Little Peter is building a stack out of his toy blocks. He is using two kinds of blocks – cubes and cylinders – and wants to stack all of them into a single tower, where each block other than the topmost block has a single block standing on it. In order for the tower to be stable, the outline of each block needs to be fully contained within the outline of the block below when looking at the tower from above (the outlines are allowed to touch). Is it possible to construct such a tower, and if so, in which order do the blocks need to be stacked?

Input

The input consists of:

1. One line with an integer n ($1 \le n \le 100$), the number of blocks.

2. *n* lines, each with the description of a block. The description consists of a string giving the type of block (cube or cylinder) and an integer *a* ($1 \le a \le 1000$) giving the size of the block – if the block is a cube then *a* is its side length, and if it is a cylinder then *a* is the radius of its base (note that the height of the cylinder does not matter).

Output

If there is no way to construct the tower, output impossible. Otherwise output *n* lines, giving the order in which to stack the blocks from top to bottom.

Sample Input 1	Sample Output 1
`3 cube 7`	`cube 7`
`cube 11`	`cylinder 5`
`cylinder`	`cube 11`
`5`	

Sample Input 2	Sample Output 2
`2 cube 5`	`impossible`
`cylinder`	
`3`	

Sample Input 3	Sample Output 3
`3 cube 4`	`cylinder 2`
`cylinder`	`cube 4`
`2 cube 4`	`cube 4`