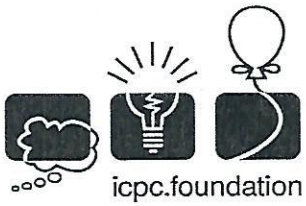


icpc International Collegiate Programming Contest

XXXIII Maratón Nacional de Programación
Colombia - ACIS / REDIS / CCPL 2019
ICPC

Problems

(This set contains 11 problems; problem pages numbered from 1 to 17)

General Information. Unless otherwise stated, the conditions stated below hold for all the problems. However, some problems might have specific requirements and it is important to read the problem statements carefully.

Program name. Each source file (your solution!) must be called

`<codename>.c`, `<codename>.cpp`, `<codename>.java`, or`<codename>.py`

as instructed below each problem title.

Input.

1. The input must be read from the standard input.
2. In most problems, the input can contain several test cases. Each test case is described using a number of lines that depends on the problem.
3. In most cases, when a line of input contains several values, they are separated by single spaces. No other spaces appear in the input. There are no empty lines.
4. Every line, including the last one, has the usual end-of-line mark.
5. If no end condition is given, then the end of input is indicated by the end of the input stream. There is no extra data after the test cases in the input.

Output.

1. The output must be written to the standard output.
2. The result of each test case must appear in the output using a number of lines, which depends on the problem.
3. When a line of results contains several values, they must be separated by single spaces. No other spaces should appear in the output. There should be no empty lines.
4. Every line, including the last one, must have the usual end-of-line mark.
5. After the output of all test cases, no extra data must be written to the output.
6. To output real numbers, if no particular instructions are given, round them to the closest rational with the required number of digits after the decimal point. Ties are resolved rounding to the nearest upper value.

A: Typing Ants

Source file name: `ants.c`, `ants.cpp`, `ants.java`, *or* `ants.py`
Author: Diego Caballero

Ants have been studying human behavior for centuries. Very recently, they have gained access to messaging apps and want to use them to communicate with colonies across the globe. Two ants have been chosen to write the first ever message: the challenge is in using the human-made keyboard efficiently.

Q	W	E	R	T	Y	U	I	O
A	S	D	F	G	H	J	K	L
Z	X	C	V	B	N	M	P	Space

The way ants move on the keyboard is simple, as they walk between two adjacent keys and press a key when needed. More precisely, in the span of one second, a single ant has three options:

1. stay at the current key,
2. move to an adjacent key (i.e., those that share a border), or
3. press the key they are currently standing on.

Both ants are allowed to freely move (option 2) and can stand at the same key at any time. However, they cannot press keys simultaneously, i.e., at most one key can be pressed during the span of one second. Since ants are known to be highly efficient and hardworking, the two chosen ants want to complete the typing task as soon as possible. In this way, they can have enough 'spare' time to go back to the colony and help with other duties.

Assuming that (i) the ants type on a keyboard layout as the one depicted above, (ii) they are as efficient as possible, and (iii) both start from the 'Space' key, what is the least number of seconds the two ants need for typing a given message?

Input

The input consists of multiple test cases. Each test case is described in single line that contains a message s ($1 \leq |s| \leq 10\,000$) to be typed by the two ants. You can assume that s contains only the characters in the keyboard layout presented in the problem description, including spaces.

The input must be read from standard input.

Output

For each test case, output the case number followed by the minimum number of seconds it will take the two ants to type s (see the sample output for details).

The output must be written to standard output.

Sample Input	**Sample Output**
UP HELLO WORLD SEND FOOD	Case #1: 6 Case #2: 23 Case #3: 22

B: Beaded Bracelets

Source file name: `bracelets.c`, `bracelets.cpp`, `bracelets.java`, *or* `bracelets.py`
Author: Camilo Rocha

Beaded bracelets are a common kids craft. Building them can improve fine motor skills with the smaller movements that occur in the wrists, hands, and fingers. On top of that, once finished, the beaded bracelets can be worn as colorful inexpensive jewelry.

Once the glamorous craft of designing beaded bracelets is mastered, the next step is to align two of them into a single two-strips beaded bracelet. Since you are still new to this craft, only beads of two colors will be considered: blue (represented by 'B') and red (represented by 'R'). In this setting, the following are descriptions of two bracelets, each with four beads:

```
RBBB        BRBR
```

Since bracelets are circular, the above descriptions are not unique. If these two beaded bracelets were to be aligned (e.g., the first one on top of the second one), then the following two-strips bracelets could be built:

```
RBBB        RBBB
BRBR        RBRB
```

However, kids these days tend to be superstitious: a two-strips beaded bracelet in which two red beads are vertically aligned is *not acceptable* (otherwise, it is *acceptable*). In the two configurations above, the one on the left is acceptable, but the second one is not acceptable because the beads in the first row are both red.

This is the actual challenge for you: given the description of two bracelets made of blue and red beads only, how many rotations of the second bracelet result in an acceptable two-strips bracelet? Note that the two bracelets must be always aligned.

Input

The input consists of several test cases. A test case comprises three lines. The first line contains an integer n ($1 \leq n \leq 100\,000$) defining the number of beads in both bracelets. The next two lines each contains the description of a blue-red beaded bracelet as a sequence made of 'B' and 'R' characters of size n.

The input must be read from standard input.

Output

For each test case output the number of rotations of the second bracelet that result in an acceptable two-strips bracelet when the first bracelet is aligned on top of the second bracelet.

The output must be written to standard output.

Sample Input	Sample Output
4 RBBB BRBR 1 R B 2 RR BR	2 1 0

C: Computerabytech Inc.

Source file name: `computer.c`, `computer.cpp`, `computer.java`, *or* `computer.py`
Author: Alejandro Sotelo

Computerabytech Inc. is a technology company specialized in the development of string algorithms. The company staff is organized in teams, each one responsible for distinct projects. Doctor Terabyte, the CEO of the company, has hired you because of your unique abilities as stringologist to lead one of the teams. As you can see, the name of the company is formed joining together the words `TERABYTE`, `TECH`, and `COMPUTER`, in such a way that the resulting name is the shortest common string containing as substrings each one of the given words:

`COMPUTERABYTECH`

Each team in the company must have a name built following the same rule, using a set of words given by Doctor Terabyte. Of course, you do not want to test manually all options to find the team's name. As a proud stringologist, you are challenged to write a program to find the minimum length possible of a string containing as substrings each one of the strings in a given set.

Input

The input consists of several test cases. The first line of a test case contains a positive integer N indicating the number of words present in the set given by Doctor Terabyte ($2 \leq N \leq 16$). Then follow N lines, each containing one single word w that consists of uppercase letters from the English alphabet ($1 \leq |w| \leq 32\,768$). You may assume that the given set does not contain repeated words.

The input must be read from standard input.

Output

For each test case, output a single line with one integer indicating the minimum length attained by a string containing as substrings each one of the strings in the set given by Doctor Terabyte.

The output must be written to standard output.

Sample Input	**Sample Output**
3 TERABYTE TECH COMPUTER 3 COMPUTER COMPANY TECHNOLOGY 5 STRING RING INGENIOUS OUSTER FIRST 2 EARTH ART	15 25 18 5

D: Dazzling Stars

Source file name: `dazzle.c`, `dazzle.cpp`, `dazzle.java`, *or* `dazzle.py`
Author: Camilo Rocha

Since the earliest days of humanity, looking up to the stars and finding meaning in their shapes has been a way of recording the history. Stars have been a great way to tell and remember stories, and have been used by farmers as a clock and by sailors as a navigation reference.

This problem is about analyzing stars; the interest is in identifying how stars are grouped in the sky. For this purpose, the following measure is defined: for any star s and positive real number r, the *r-clustering coefficient of s* is the number of stars (other than s) that are at most at distance r from s. Note that one star can have many clustering coefficients, depending on the choice of r.

Given a 2D description of the starscape, you are required to design an algorithm for computing the clustering coefficients of stars with respect to the 2D (squared) Euclidian distance.

Input

The input consists of several test cases. Each test case begins with two blank-separated integers N and Q ($1 \leq N, Q \leq 10\,000$) representing, respectively, the number of stars and the number of queries. Then follow N lines; the n-th line ($1 \leq n \leq N$) contains two blank-separated integer numbers x_n and y_n ($0 \leq |x_n|, |y_n| \leq 5\,000$) indicating the Cartesian position (x_n, y_n) of star n in the starscape. Then follow Q lines, each containing two blank-separated integer numbers s and r ($1 \leq s \leq N$ and $0 \leq r < 10\,000$). You can assume that all stars are located at different positions.

The input must be read from standard input.

Output

For each query, output the r-clustering coefficient of star s in a single line. Output a star '*' between any two consecutive test cases.

The output must be written to standard output.

Sample Input	Sample Output
5 3 0 0 -1 3 4 1 1 3 2 2 1 2 1 4 2 2 1 1 0 0 1 1	0 3 1 * 0

E: Grade Estimate

Source file name: `estimate.c`, `estimate.cpp`, `estimate.java`, *or* `estimate.py`
Author: Rafael García

In his first programming course homework, Alex was given the following challenging task: given a positive integer n, he has been asked to design and implement an algorithm to find three integer numbers p, q, r such that $n = p \cdot q \cdot r$ and to print the value of $\frac{1}{n}$.

By now, Alex has submitted his solution to the homework. However, he has realized that in order to compute the value of $\frac{1}{n}$, his algorithm uses the following equality:

$$\frac{1}{n} = \frac{1}{p} + \frac{1}{q} + \frac{1}{r}.$$

This is indeed a problem because his algorithm works correctly depending on the choice of n. For example, if $n = 6$ the algorithm finds $p = 1$, $q = -2$ and, $r = -3$, and computes

$$\frac{1}{6} = \frac{1}{1} + \frac{1}{(-2)} + \frac{1}{(-3)}.$$

The final grade for this homework will be given as an integer number in the range 0..100: it corresponds to the percentage of cases correctly answered by each submission. Given the input that will be used to grade the homework, can you help Alex to estimate the grade of his submission?

Input

The input consists of several test cases. A test case is described by a line with m ($0 < m \leq 25$) positive integer numbers representing the set of numbers for evaluating Alex's task. Each number n in this list satisfies $0 < n \leq 82\,945\,319\,184$.

The input ends with a line containing the character 0.

The input must be read from standard input.

Output

For each test case, output one line containing an integer number corresponding to the integral part of the percentage of the m cases correctly answered by Alex's algorithm.

The output must be written to standard output.

Sample Input	Sample Output
5 6 7 42 6 3 10 20 1 1 3 6 42 82945319184 0	33 66 0 100

F: Fibcod Cryptocurrency

Source file name: `fibcod.c`, `fibcod.cpp`, `fibcod.java`, *or* `fibcod.py`
Author: Juan Camilo Corena

The Fibcod Cryptocurrency is the most popular payment method nowadays among participants of programming contests. To use it, participants are given a unique wallet number to send and receive money anywhere in the world, without paying any transfer fees!

As expected, assigned wallet numbers need to be unique. For this reason, Fibcod uses a huge collection of wallet numbers: they are large sequences made of the digits 0 and 1. However, unlike binary numbers where the digit 1 indicates that a given power of 2 needs to be added, the digit 1 in a Fibcod wallet number indicates which numbers in the following (shifted) Fibonacci sequence need to be added:

$$F_0 = 1$$
$$F_1 = 2$$
$$F_n = F_{n-1} + F_{n-2}, \quad \text{for } n \geq 2.$$

For example, the wallet number 11101 interpreted in base-10 is $17 = 8 + 5 + 3 + 1$.

Recently, one of the Fibcod Cryptocurrency founders noticed that different wallet numbers, when converted to base-10, identify the same wallet. For instance, 17 is also represented by the wallet number 100101 (i.e., $17 = 13 + 3 + 1$). Realizing that this goes against Fibcod's motto 'One for one and zero for none' and is likely to be the source of ugly legal disputes in the future, the cryptocurrency founders have agreed --in a bold and unprecedented decision-- to ban the assignment of a wallet number if its base-10 interpretation has been assigned before.

Your task is to design and implement an efficient algorithm to convert Fibcod's wallet numbers to base 10, so that it can later be integrated to the wallet number assignment verification system.

Input

The input consists of several test cases, each comprising a Fibcod wallet number described in two lines. The first line contains a number $1 \leq n \leq 1\,000$ indicating the number of blocks of repeated 1s and 0s that concatenated define the wallet number. The second line contains n integer numbers $b_1, b_2, \ldots, b_n$ indicating how many times a given digit should be repeated per block: a b_i with odd index i, indicates a number of repeated 1s; a b_i with even index i, indicates a number of repeated 0s (for odd i, $1 \leq b_i \leq 100$; for even i, $1 \leq b_i \leq 10^9$). For example, in this format, the Fibcod wallet number 11000011111 is the concatenation of 3 blocks, where $b_1 = 2$ (i.e., 11), $b_2 = 4$ (i.e., 0000), and $b_3 = 5$ (i.e., 11111).

The input ends with a line containing 0 (which should produce no output).

The input must be read from standard input.

Output

For each test case, output the base-10 value of the given Fibcod's wallet number modulo 524 288.

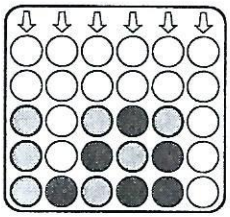
The output must be written to standard output.

Sample Input	**Sample Output**
3 3 1 1 3 2 4 5 3 2 11 5 7 100 100000 100 100000 100 100000 100 0	17 252 6784 375403

G: Game Conundrum

Source file name: `game.c`, `game.cpp`, `game.java`, *or* `game.py`
Author: Nicolás Cardozo

Alice and Carl have a mean competitive siblings' rivalry. During summer, they play board games and declare an overall winner for the whole year. The summer holidays are about to end and they are tied with one final session to play in the family' favorite ''4 In A Line''. In this game, the two players have tokens of two different colors (Alice one color and Carl another color) and take turns to place them in a vertically set-up board of $n \times n$ size. Tokens are inserted from the top of a column and fall into the lowest unoccupied position in that column. The first player to complete a straight (horizontal, vertical, or diagonal) line of 4 tokens of its color wins the game.



At some point during the last session, you hear the hullabaloo of the summer. There is a big argument coming from the family's game room. When you arrive, you see the ''4 In A Line'' board on the floor and all the tokens played during the session scattered around. One of Alice and Carl, a sore loser, intentionally threw the board to the floor when the other player won the game. The thing is that both, Alice and Carl, are claiming the victory for the session and thus are the self-proclaimed overall winners of the year.

Given the number of tokens played by each one of Alice and Carl during the final session, and the name of who played the first token, you are to solve the game conundrum: who won the last session of the game?

Input

The first line of the input contains the number of test cases ($1 \leq g \leq 100$). Each test case consists of three lines describing a game. The first line of a game contains the side n of the board game ($4 \leq n \leq 128$). The second line contains two blank-separated integers a and c ($4 \leq a, c \leq 8\,192$) representing the number of tokens played by Alice and Carl, respectively. The third line contains the name of the player that started the game, either `'Alice'` `'Carl'`.

The input must be read from standard input.

Output

For each test case, output one line with the name of the winner of the last session: `'ALICE'` if Alice is the winner and `'CARL'` if Carl is the winner (ignore the quotes).

The output must be written to standard output.

Sample Input	Sample Output
2 5 5 4 Alice 9 20 20 Carl	ALICE ALICE

H: Silos of Hernando

Source file name: `hernando.c`, `hernando.cpp`, `hernando.java`, *or* `hernando.py`
Author: Rafael García

Hernando is an expert silo architect who was commissioned to design and build a collection of silos for storing various grains. Each silo is a cylinder, and is described by the radius and center of its circular base, and its height. The silos designed by Hernando are all equal (i.e., have the same radius and height) and are in operation.

The next task is to enclose all the silos behind an electrified fence. The design needs to guarantee that the fence is at least one meter away from each silo. Assuming that the fence will have the same height as the silos and that no material goes to waste, how many square meters of material does Hernando need to build the fence?

Input

The input consists of several test cases. A case begins with a line containing three positive integer numbers n ($0 < n \leq 10\,000$), r ($0 < r \leq 1\,000$), and h ($0 < h \leq 100$) denoting the number of silos, the radius (in meters), and the height (in meters) of each silo, respectively. Then follow n lines, each one with two blank-separated positive integer numbers x and y ($0 < x, y \leq 10\,000$), describing the center of a silo as a point (x, y) in the Cartesian plane. It can be assumed that the silos in a test case are pairwise disjoint.

The input must be read from standard input.

Output

For each test case, output one line with the area of the fence in square meters (with 6 decimal places).

The output must be written to standard output.

Sample Input	Sample Output
4 1 1 1 1 3 1 1 3 3 3 4 1 10 1 1 3 1 1 3 3 3 7 20 11 0 300 200 300 100 100 200 100 300 0 0 0 300 300	20.566371 205.663706 14651.415806

I: Indexing App

Source file name: `index.c`, `index.cpp`, `index.java`, *or* `index.py`
Author: Rafael García

The Colombian Competitive Programming and aLgorithms (CCPL) book --arguably better than D. Knuth's famous volumes-- is about to go on sale worldwide. After more than 15 years of careful writing and editing, it includes a significant number of volumes comprising pseudocode, analysis, problems, and references!

There are so many volumes and the information is so varied that the editor has decided to use an app for the topics' index. The idea is to offer the CCPL's indexing app from the most popular app stores, free of charge (no spam either!): given a text fragment t from the book (algorithms and equations included), it will indicate each of the occurrences of t in the book.

For testing and validating purposes, the prototype of the app only needs to find the number of the first volume in which the fragment t occurs. Can you help in implementing the first CCPL's indexing app prototype?

Input

The input consists of several test cases. A case begins with a line containing two blank-separated integers N and Q ($0 < N < 100$ and $0 < Q \leq 1\,000$), where N is the number of volumes and Q is the number of texts that must be located in the volumes. Then N lines follow describing the volumes of the book, one volume per line: a volume is a text V with $0 < |V| \leq 1\,000\,000$. Finally, Q lines follow describing the texts to find, one text t per line ($0 < |t| \leq 1\,000$). The end of input is given by a line containing two blank-separated zeroes.

The input must be read from standard input.

Output

For each test case, output Q lines, one for each text: if the text is in the book, then output the number of the first volume where the text occurs (starting from 1); otherwise, output the symbol 'n' (ignore the quotes).

The output must be written to standard output.

Sample Input	Sample Output
2 4 Foundations.TheRoleOfAlgorithmsInComputing.GettingStarted.Theorem. SortingAlgorithms.TheHeapsort.Quicksort. Algorithms sort Theorem Programming 3 3 DynamicProgrammingRodcuttingMatrix-chainmultiplication ElementsofdynamicprogrammingLongestcommonsubsequenceOptimalsearch HashTablesDirect-addresstablesHashtablesHashfunctionsPerfecthashing Programming hash Theorem 0 0	1 2 1 n 1 3 n

J: Justice League Counter-strategy

Source file name: `justice.c`, `justice.cpp`, `justice.java`, *or* `justice.py`
Author: Germán Sotelo

Even with its might and power, the Justice League has struggled with crime in Gotham City since villains always find clever ways to win their encounters. Lately, whenever the superheroes attack any villain, the villains create alliances and end up defeating the good guys.

To counter this strategy, the superheroes have devised a counter-strategy: during one night, all the villains will be simultaneously attacked by superheros to prevent the formation of alliances. However, the superheroes have not decided which one will fight which villain. To make the decision, they have collected stats on the *attack power* of every superhero and the *defense level* of every villain. They know that a villain can be defeated in an encounter whenever his/her defense level is below the attack power of his/her superhero rival. The counter-strategy is to have 1-on-1 encounters: each villain is to be defeated by exactly one superhero during the whole night.

Your help has been requested at the Hall of Justice. Given the stats of every superhero and villain in Gotham City, you have been challenged to find the largest possible number of villains that can be defeated by the counter-strategy. Since it will be a surprise attack, you can assume that villains will not have time to assemble into teams to defend themselves. Take into account that during that night each superhero can have at most one encounter with a villain.

Input

The input consists of several test cases. Each test case begins with a line containing two blank-separated integers s and v ($0 < s, v < 10\,000$) indicating the number of superheroes and villains, respectively. Each of the following s lines has one positive integer a, the attack power of a superhero ($0 < a < 100\,000$). Each of the following v lines has one positive integer d, the defense level of a villain ($0 < d < 100\,000$).

The input ends with a line containing two blank-separated zeroes.

The input must be read from standard input.

Output

For each test case, output a single line with the largest possible number of defeated villains.

The output must be written to standard output.

Sample Input	Sample Output
1 1 200 100 1 2 200 100 100 0 0	1 1

K: Kitchen Mess

Source file name: `kitchen.c`, `kitchen.cpp`, `kitchen.java`, *or* `kitchen.py`
Author: Germán Sotelo

Remy is an intelligent mouse with a highly developed sense of taste; he has become one of the most renowned chefs in Paris. However, his popularity has caused a lot of skepticism between human chefs and he has started to receive challenges to test his abilities.

Recently, Remy received a challenge from chef Gusteau to compete for dessert: the best tiramisu recipe. Even though Remy is a skilled chef, he has never focused on desserts. Moreover, he does not even have the right tools for deserts. He has asked Alfredo, a friend who owns a bakery, to lend him the bakery kitchen to try out recipes. Alfredo accepts this request under two conditions: the kitchen can only be used at night, and the next morning it should be completely clean and tidied up. This also means that all baking trays are required to be stacked up in the least possible number of piles. All baking trays are rectangular with various lengths and widths: a tray can only be stacked on top of another tray that has equal or larger dimensions.

One night, after trying out many times, Remy finally discovered the perfect tiramisu recipe. However, he was so tired (energy in mice can somehow be easily depleted) that after washing all bakery trays, night caught up with him and felt asleep. Next morning, he was waked up by the sounds of people coming to open the bakery. Quickly, Remy tried to put everything into place, but one thing was left: to pile up the baking trays according to the rule of the pastry chef.

Remy needs help: can you find the least possible number piles in which all the baking trays can be put complying with Alfredo's rule.

Input

The input consists of several test cases. Each test case begins with a line containing one integer n ($1 < n \leq 1\,000$), the number of baking trays available in the kitchen. Each of the following n lines describes the dimensions of each baking tray: two blank-separated integers x and y indicating the width and length of the tray ($1 \leq x \leq 1\,000$, $1 \leq y \leq 1\,000$).

The input must be read from standard input.

Output

For each test case, output a single line with the least number of piles that can be formed using all the baking trays under the given condition.

The output must be written to standard output.

Sample Input	**Sample Output**
3 3 3 2 1 1 2 3 3 3 2 2 9 1	1 2