Problem A
Circuit Math

You are enrolled in the Computer Organization and Architecture course at your university. You decide to write a program to help check your work by computing the output value of a combinational digital circuit, given its inputs.

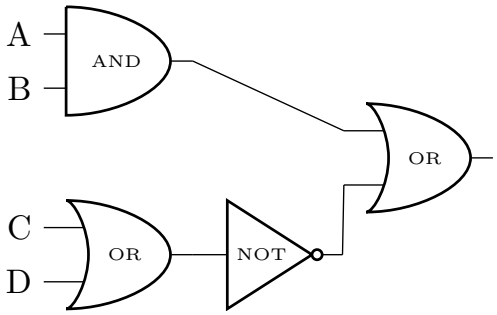

Figure A.1: An example of a combinational digital circuit. See the text for an explanation.

Consider the circuit shown in Figure A.1, which we use for illustration. This circuit has four inputs (letters A through D on the left), each of which is either true or false. There are four 'gates' each of which is one of three types: AND, OR, or NOT. Each gate produces either a true or false value, depending on its inputs. The last gate (the OR on the right) produces the output of the entire circuit. We can write these three types of gates in text by their equivalent *logical operators*: $\star$ for AND, + for OR, and − for NOT. In what follows, we'll use the operators rather than gates to describe circuits.

Here is how these operators work. Given an assignment of true (`T`) or false (`F`) for each input, the operators produce the truth value indicated in the following tables:

A	B	A B $\star$	A B +
T	T	T	T
F	T	F	T
T	F	F	T
F	F	F	F

A	A −
T	F
F	T

Notice that AND and OR take two inputs, whereas NOT operates on only one input. Also, we use *postfix notation* to write expressions involving operators (like `A B` $\star$), where the operator comes *after* its input(s) (just as how in Figure A.1, each gate in the circuit diagram comes after its inputs).

When we describe a valid circuit in postfix notation, we use the following syntax.

- An uppercase letter (`A` through `Z`) is a valid circuit. In other words, an input alone (without any gates) is a valid circuit (which produces as output its own input value).
- If `<C1>` and `<C2>` are valid circuits, then '`<C1> <C2>` $\star$' is a valid circuit that produces the AND of the outputs of the two subcircuits.
- If `<C1>` and `<C2>` are valid circuits, then '`<C1> <C2>` +' is a valid circuit that produces the OR of the outputs of the two subcircuits.

- If <C1> is a valid circuit, then '<C1> -' is a valid circuit that produces the NOT of <C1>'s output.

No other description is a valid circuit.

Thus, one of the ways the circuit in Figure A.1 could be described using postfix notation is as the string:

A B * C D + - +

Given a truth value (T or F) for each of the inputs (A, B, C, and D in this example), their values propagate through the gates of the circuit, and the truth value produced by the last gate is the output of the circuit. For example, when the above circuit is given inputs A=T, B=F, C=T, D=F, the output of the circuit is F.

Given an assignment to variables and a circuit description, your software should print the output of the circuit.

Input

The first line of the input consists of a single integer n, satisfying $1 \leq n \leq 26$, denoting the number of input variables. Then follows a line with n space-separated characters. Each character is either T or F, with the ith such character indicating the truth value of the input that is labeled with the ith letter of the alphabet.

The last line of input contains a circuit description, which obeys the syntax described above. Each circuit is valid, uses only the first n letters of the alphabet as input labels, and contains at least 1 and at most 250 total non-space characters.

Note that while each variable is provided only one truth value, a variable may appear multiple times in the circuit description and serve as input to more than one gate.

Output

Print a single character, the output of the circuit (either T or F), when evaluated using the given input values.

Sample Input 1	Sample Output 1
4 T F T F A B * C D + - +	F

Problem B
Diagonal Cut

Photo of a chocolate being cut. Image from Simone_ph at pixabay.com

Quido and Hugo are making a chocolate cake. The central ingredient of the cake is a large chocolate bar, lying unwrapped on the kitchen table. The bar is an $M \times N$ rectangular grid of chocolate blocks. All of the MN blocks are rectangles of identical shape and size. The chocolate bar is of top quality and the friends want to eat part of it, before the rest is used in the cake.

"OK," says Quido, "let's divide the whole bar into two triangular chunks by a straight diagonal cut from its upper-left corner to its lower-right corner. We will then eat all of the blocks which have been cut exactly in half, into two equal-area pieces. You will eat one half and I will eat the other half of each such block. All other blocks, that is, the blocks which are either uncut or cut into two parts of different sizes, will go directly into the cake. Of course, we will make sure the cut is perfectly precise.

Let's see how much chocolate we get to eat!"

Input

The input consists of two space-separated integers M and N given on a single line, (where $1 \leq M, N \leq 10^{18}$). The numbers M and N denote the number of blocks in one column and in one row, respectively, in the chocolate bar.

Output

Print the number of blocks of the chocolate bar which are cut into exactly two pieces of equal area.

Sample Input 1	Sample Output 1
6 10	2

Sample Input 2	Sample Output 2
75206452536745713 10322579177493903	40318322589

This page is intentionally left blank.

Problem C
Gerrymandering

The original political cartoon about Gerrymandering. Image from Wikipedia.

Electoral systems across the world can vary widely. In some systems, such as *winner-take-all*, the winner is determined by the plurality of votes—the candidate that receives the most votes wins, and the loser(s) do not get a position.

Such elections can have "wasted votes." Conceptually, a wasted vote is a vote that did not affect the election outcome. While the exact definition of a wasted vote varies, we'll adopt the following definition: in an election with V voters, every vote for a losing candidate is wasted (these are called *lost votes*), and every vote for a winning candidate beyond the strict majority of $\lfloor V/2 \rfloor + 1$ votes the candidate needs to win is wasted (these are called *excess votes*). For this problem we'll consider a two-party system (let's call the parties A and B) with elections that always involve one candidate from each party.

Let's illustrate wasted votes with a simple example between two candidates in a district. Suppose that the candidate for party A receives 100 votes and the candidate for party B receives 200 votes. All 100 votes for party A are wasted (lost votes for A), and 49 votes for party B are wasted (excess votes for B). This is because B needs 151 ($\lfloor (100 + 200)/2 \rfloor + 1$) votes to win (over A), so the remaining 49 are wasted.

Political scientists use wasted votes to compute the *efficiency gap*, a single number that summarizes wasted votes. Suppose we have a number of races in different districts, where each district elects one person. Across all districts there are V total votes cast, with w_A total wasted votes for party A and w_B total wasted votes for party B. Then the efficiency gap is:

$$E(V, w_A, w_B) = \frac{|w_A - w_B|}{V}.$$

A low efficiency gap indicates that the elections are competitive, and that the number of candidates elected from each party is representative of the total voting share for each party. When the efficiency gap is high, this can be an indication of *gerrymandering*. Gerrymandering refers to organizing voting districts in a way that favors a particular political outcome. Two common ways of doing this are to "pack" similar voters into districts, or "crack" them across multiple districts; both ways tend to diminish those voters' influence in electing candidates they would like to win.

In an election, districts are made up of precincts. A precinct is an indivisible group of voters. The votes for all precincts in a district are added together to find the results for that district. In this problem you are given a description of a number of precincts: the party vote totals for each precinct, and how those precincts have been grouped into districts. For each district, determine the party that wins and the wasted votes for each party. Then determine the efficiency gap between the two parties over all the districts.

Input

The input describes one election. The first line contains two integers P and D, where $1 \leq P \leq 10\,000$ and $1 \leq D \leq \min(1\,000, P)$. These indicate, respectively, the number of voting precincts and districts. Following this are P lines describing the precincts. Line i contains 3 numbers: the district d_i that precinct i is assigned to ($1 \leq d_i \leq D$), the number of votes for the candidate from party A ($0 \leq a_i \leq 100\,000$), and the number of votes for the candidate from party B ($0 \leq b_i \leq 100\,000$). It is guaranteed that:

- for each precinct i, $0 < a_i + b_i$,
- each district is assigned at least one precinct, and
- there are no ties within any district.

Output

For each of the districts from 1 to D, print which party wins (a single character, either `A` or `B`). Then print the number of wasted votes for party A and for party B, in order. Finally, after reporting on all the districts, print the efficiency gap as measured over all the districts. The efficiency gap should be accurate to within an absolute error of 10^{-6}.

Sample Input 1

```
5 3
1 100 200
2 100 99
3 100 50
3 100 50
2 100 98
```

Sample Output 1

```
B 100 49
A 1 197
A 49 100
0.1965897693
```

Sample Input 2

```
4 4
3 100 99
2 100 99
1 100 99
4 100 99
```

Sample Output 2

```
A 0 99
A 0 99
A 0 99
A 0 99
0.4974874372
```

Sample Input 3

```
4 4
4 99 100
1 100 99
3 100 99
2 99 100
```

Sample Output 3

```
A 0 99
B 99 0
A 0 99
B 99 0
0.0000000000
```

Problem D
Missing Numbers


Photo by Valerie Everett.

You enjoy your new job as a teacher of young children. It's fun to see them learning to count, recognize letters, draw, and interact with the world.

One common problem you've noticed is that children often forget numbers when counting. For example, early on they might count "one, two, three, five, six." You have to remind them about that "four" that they didn't say. And as they get more proficient and clever, they may use the "quick" way of counting: "one, two, skip a few, ninety-nine, one hundred!"

Please write a program that can help you (and your students) identify the missing numbers when they are counting.

Input

The first line of input contains a single integer n, where $1 \leq n \leq 100$. Each of the next n lines contains one number that the child recited. Each recited number is an integer between 1 and 200 (inclusive). They are listed in increasing order, and there are no duplicates.

Output

If the child recited all the numbers between 1 and the last number they recited, then print `good job`.

If the child missed any numbers between 1 and the last number they recited, then print those missing numbers in increasing numeric order, one per line.

Sample Input 1	Sample Output 1
9 2 4 5 7 8 9 10 11 13	1 3 6 12

Sample Input 2	Sample Output 2
5 1 2 3 4 5	good job

Problem E
NVWLS


The Kryptos statue, Langley, VA. Source Wikipedia

NVWLS, or "No Vowels" puzzles are popular among puzzle enthusiasts. For example, consider the following no-vowels message:

```
BTWNSBTLSHDNGNDTHBSNCFLGHTLSTHNNCFQLSN
```

which is inscribed on the famous "Kryptos" statue located at the CIA's headquarters in Virginia. This message is derived from the following sentence by removing all vowels and spaces:

```
BETWEEN SUBTLE SHADING AND THE ABSENCE OF LIGHT
LIES THE NUANCE OF IQLUSION
```

Given a dictionary (a set of words that can be used to construct a sentence) and a message (which comes from a sentence which uses only those words, but with all vowels and spaces removed), reconstruct the original sentence using the dictionary words!

Input

The first line contains an integer n denoting the number of words in the dictionary. The next n lines each contain a dictionary word using one or more uppercase English letters. Each word contains at least one consonant.

For the purposes of this problem, the letters `A`, `E`, `I`, `O`, and `U` are vowels and all other letters are consonants.

The dictionary is followed by a single non-empty line of uppercase consonants representing the no-vowels message. It is guaranteed that the no-vowels message can be constructed in at least one way using only the dictionary words.

The total number of letters in all dictionary words is no greater than $100\,000$. The total number of letters in the no-vowels message does not exceed $300\,000$.

Output

Output a whitespace-separated sequence of dictionary words that yields the original no-vowels message when all spaces and vowels are removed. If there are multiple reconstructions, choose the one with the largest overall number of vowels. If there are still multiple reconstructions, you may output any one of them. No judge input will require your program to output more than $15\,000\,000$ characters.

Sample Input 1

```
11
BETWEEN
SUBTLE
SHADING
AND
THE
ABSENCE
OF
LIGHT
LIES
NUANCE
IQLUSION
BTWNSBTLSHDNGNDTHBSNCFLGHTLSTHNNCFQLSN
```

Sample Output 1

```
BETWEEN SUBTLE SHADING AND THE ABSENCE OF LIGHT LIES THE NUANCE OF IQLUSION
```

Sample Input 2

```
4
NA
NNANNA
NANNA
BATMAN
NNNNNNNNNNNNNBTMN
```

Sample Output 2

```
NA NA NA NA NA NA NA NA NA NA NA NA NA BATMAN
```

Problem F
Prospecting

Prospectin' Pete has a lead on a new titanium mine, and needs your help pitching a mining operation to investors. The mine can be represented as a tree: the mine entrance is the root of the tree, other tree nodes are pockets of underground titanium ore, and the tree edges are potential tunnels Pete can dig between two pockets (or between the mine entrance and one pocket, for edges adjacent to the root node). The tunnel connecting the i-th ore pocket to its parent has a length of y_i feet. One of the tree leaves contains the motherlode, and each other ore pocket contains x_i dollars worth of ore.

Pete begins at the mine entrance, and his goal is to reach the motherlode. Obviously, Pete cannot travel to pocket i until all y_i feet of dirt in the tunnel leading to it has been removed. But once a tunnel has been completely excavated, Pete can travel that tunnel, in both directions, as many times as he wants.

Pete explores the mines by repeatedly performing the following steps, in order:

1. Pete chooses a tunnel that he can reach from his current location, and that has not yet been fully excavated.
2. Pete digs through one foot of the chosen tunnel; this costs him one dollar.
3. If the tunnel is now completely clear, Pete travels through the tunnel to the newly opened pocket and mines the ore. If that pocket contains the motherlode, then he stops exploring the mine. Otherwise, he sells the ore in that pocket for x_i dollars and continues exploring.

Note that the tunnel Pete chooses to dig in the first step of each round of digging does not need to be adjacent to his current location, so long as Pete can reach the tunnel by traveling through the mine along a sequence of completely-excavated tunnels. He can also choose a different tunnel each round, even if the tunnel he was digging in the previous round is not yet completely excavated. If Pete ends a round of digging with no money, and without having reached the motherlode, the mining operation is a bust and Pete goes home bankrupt.

Pete has surveyed the geology of the area and knows the mine layout, as well as the amount of ore in each chamber, but hasn't yet decided on a strategy for how to dig the tunnels. He knows that in addition to any riches he earns from the mine itself, he will need some amount of startup money in order to reach the motherlode, and so he is courting investors. He wants to present them with two pieces of information:

- The minimum number of dollars a that Pete must begin with in order for his venture to be successful even in the worst-case: for it to be guaranteed that he reaches the motherlode no matter how he chooses the tunnel to excavate during each round of digging.
- The minimum number of dollars b that Pete must begin with in order to reach the motherlode in the best-case scenario where Pete chooses tunnels optimally.

This information will allow the investors to decide how much to fund Pete, based on how much they trust him to dig optimally without making any mistakes.

Given the mine layout, compute a and b.

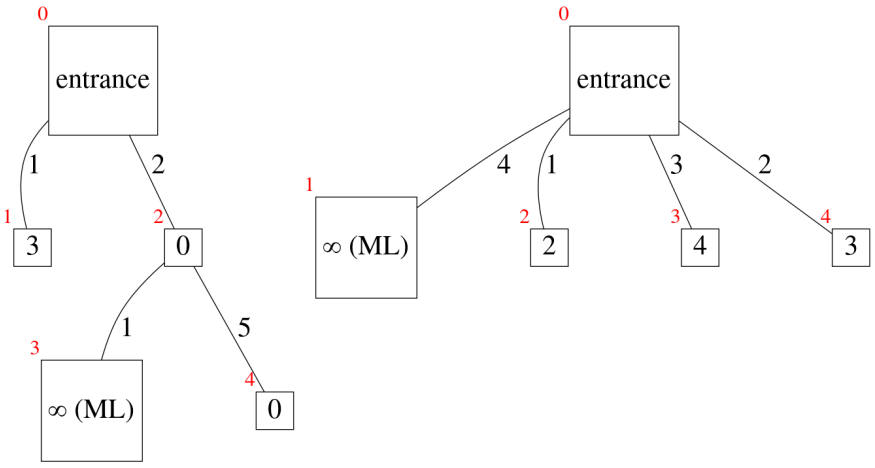

Figure F.1: Illustrations of sample inputs 1 (on the left) and 2. Edges represent tunnels and nodes represent ore pockets. Ore values x_i in each pocket and length y_i of each tunnel are written in black text ("ML" stands for the motherlode). Nodes are labeled in red with their indices.

Input

The first line of input contains an integer n ($2 \leq n \leq 200\,000$), the number of nodes in the tree representing Pete's mine. The remaining input consists of $n - 1$ lines, the ith of which (starting at 1) contains three integers p_i, x_i, and y_i encoding information about the ith node of the tree. p_i ($0 \leq p_i < i$) is the index of the ith node's parent in the tree. A parent index of zero means that the node's parent is the mine entrance (root of the tree). x_i is the value of the ore in node i, and y_i ($1 \leq y_i \leq 1\,000$) is the length of the tunnel connecting node p_i to node i. Exactly one node has $x_i = -1$, indicating that the node contains the motherlode; all other inputs satisfy $0 \leq x_i \leq 1\,000$.

Note that node zero is the mine entrance; it contains no ore and has no corresponding line in the input. You may assume that the motherlode is a leaf of the tree (no node has the motherlode node as a parent).

Output

Print two space-separated integers a and b, the worst-case and best-case investment Pete needs in order to clear a path to the motherlode, as described in more detail above.

Sample Explanation

Consider the mine described in sample input 1 and illustrated in Figure F.1. Pete needs 8 dollars to guarantee he reaches the motherlode even with poor digging choices: in the worst-case scenario he spends 2 dollars to completely clear the tunnel leading into pocket 2, and then 5 more dollars to clear the tunnel leading to pocket 4. With his last dollar he either digs his way to the motherlode, or opens the tunnel to pocket 1, which gives him enough money to reach the motherlode in the next round of digging. In the best-case scenario he needs only one dollar: he spends that dollar to first dig through the tunnel leading into pocket 1, and with the 3 dollars of ore he mines there, he can dig through the two tunnels on the path from the entrance to the motherlode.

A second example is provided in sample input 2 which is also illustrated in Figure F.1. In one worst-case scenario, Pete excavates three feet of the tunnel leading to pocket 1, two feet of the tunnel leading to pocket 3, and one foot of the tunnel leading to pocket 4, all without mining any ore. This costs him 6 dollars. With a seventh dollar, Pete is guaranteed to tunnel into an ore pocket which finances exploration of the rest of the mine (including the motherlode). In the best-case scenario, Pete needs only a single dollar: he first digs to pocket 2, then 4, then 3, then 1, finding the motherlode.

Sample Input 1

```
5
0 3 1
0 0 2
2 -1 1
2 0 5
```

Sample Output 1

```
8 1
```

Sample Input 2

```
5
0 -1 4
0 2 1
0 4 3
0 3 2
```

Sample Output 2

```
7 1
```

This page is intentionally left blank.

Problem G
Research Productivity Index

Angela is a new PhD student and she is nervous about the upcoming paper submission deadline of this year's research conference. She has been working on multiple projects throughout the past year. Luckily most of the projects concluded successfully, and she came up with n candidate papers. However not all of the papers were born equal—some have better results than others. Her advisor believes she should only submit the papers with "good enough" results so they have a high chance of getting accepted.

Angela's research group has a unique way of evaluating the success of paper submissions. They use the *research productivity index*, defined as $a^{a/s}$, where s is the total number of papers submitted, and a is the number of papers that are accepted by the conference. When $a = 0$, the index is defined to be zero. For example:

- if one paper is submitted and it gets accepted, the index is $1^{1/1} = 1$;
- if 4 papers are submitted and all get accepted, the index is $4^{4/4} = 4$;
- if 10 papers are submitted and 3 get accepted, the index is $3^{3/10} \approx 1.390389$;
- if 5 papers are submitted and 4 get accepted, the index is $4^{4/5} \approx 3.031433$;
- if 3 papers are submitted and all get rejected ($a = 0$), the index is 0.

Intuitively, to get a high research productivity index one wants to get as many papers accepted as possible while keeping the acceptance rate high.

For each of her n papers, Angela knows exactly how likely it is that the conference would accept the paper. If she chooses wisely which papers to submit, what is the maximum expected value of her research productivity index?

Input

The first line of the input has a single integer n ($1 \leq n \leq 100$), the number of Angela's candidate papers. The next line has n space-separated integers giving the probability of each paper getting accepted. Each probability value is given as an integer percentage between 1 and 100, inclusive.

Output

Output the maximum expected value of Angela's research productivity index. Your answer is considered correct if it has an absolute or relative error of no more than 10^{-6}.

Sample Input 1	Sample Output 1
5 30 50 70 60 90	2.220889579

Sample Input 2	Sample Output 2
6 30 90 30 90 30 90	2.599738456

Sample Input 3	Sample Output 3
4 10 10 10 10	0.368937005

Problem H
Running Routes

The administrators at Polygonal School want to increase enrollment, but they are unsure if their gym can support having more students. Unlike a normal, boring, rectangular gym, the gym floor at Polygonal is a regular n-sided polygon! They affectionately refer to the polygon as P.

The coach has drawn several running paths on the floor of the gym. Each running path is a straight line segment connecting two distinct vertices of P. During gym class, the coach assigns each student a different running path, and the student then runs back and forth along their assigned path throughout the class period. The coach does not want students to collide, so each student's path must not intersect any other student's path. Two paths intersect if they share a common point (including an endpoint).

Given a description of the running paths in P, compute the maximum number of students that can run in gym class simultaneously.

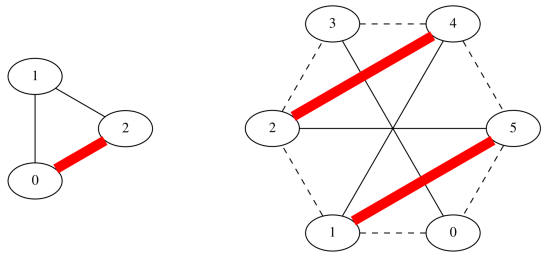

Figure H.1: Illustrations of the two sample inputs, with possible solutions highlighted in thick red lines. Solid black lines represent running paths that are not assigned to a student, and dashed black lines are used to show the boundary of P in places where no running path exists.

Input

The first line contains an integer n ($3 \leq n \leq 500$), the number of vertices in P. (The vertices are numbered in increasing order around P.) Then follows n lines of n integers each, representing a $n \times n$ symmetric binary matrix which we'll call M. The j^{th} integer on the i^{th} line M_{ij} is 1 if a running path exists between vertices i and j of the polygon, and 0 otherwise. It is guaranteed that for all $1 \leq i, j \leq n$, $M_{ij} = M_{ji}$ and $M_{ii} = 0$.

Output

Print the maximum number of students that can be assigned to run simultaneously on the running paths, given the above constraints.

Sample Input 1

```
3
0 1 1
1 0 1
1 1 0
```

Sample Output 1

```
1
```

Sample Input 2

```
6
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 1 1
1 0 0 0 0 0
0 1 1 0 0 0
0 1 1 0 0 0
```

Sample Output 2

```
2
```

Problem I
Slow Leak


TheDigitalWay

You are an avid cyclist and bike every day between your home and school. Usually, your ride is uneventful and you bike along the shortest path between home and school. After school this afternoon you realized you have a slow leak in your bike tire—the tire can hold some air, but not for long. Refilling the tire allows you to ride your bike for some distance, after which your tire goes flat again and it becomes impossible to ride any further (and you refuse to walk your bicycle).

Luckily for you, your city has installed several bike repair stations at intersections throughout town where you can refill your tire and bike again until the tire goes flat. There's a repair station at your school too, so that you can fill up your tire before you start on your trip home.

You've calculated how far you can bike before your tire runs out of air and you know the layout of your town, including all the intersections, distances between them, and the locations of the repair stations. What is the shortest possible trip from school to your home that you can take without becoming stuck due to a flat tire? (You do not become stuck if you roll into a repair station, or your home, at the exact same time as your tire goes flat.)

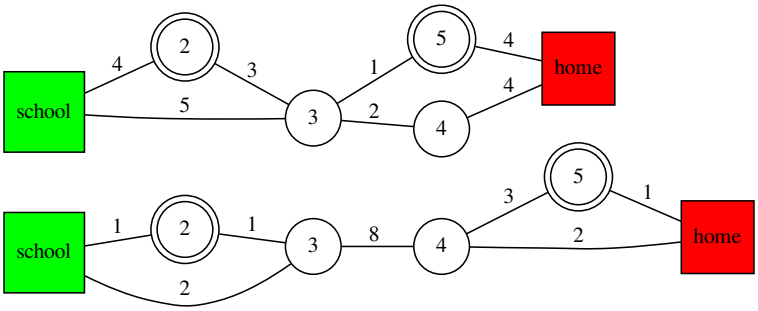

Figure I.1: An illustration of the two sample inputs.

Input

The first line of input contains four integers n, m, t, and d, satisfying $2 \leq n \leq 500$, $0 < m \leq n(n-1)/2$, $0 < t < n$ and $0 < d < 2^{31}$. The value n represents the number of intersections in the city, m represents the number of roads in the city, t is the number of repair stations and d is the distance that you can bike (starting with a fully inflated tire) before your tire goes flat again. The intersections are numbered from 1 to n. Your school is at intersection 1 and your home is at intersection n.

The second line of input contains t integers, with values between 2 and $n-1$, inclusive. These are the intersections

where the repair stations are located (excluding your school's repair station). All integers on this line are unique.

The next m lines represent the roads in your town. Each has three integers i, j, k, where $1 \leq i < j \leq n$, and $0 < k < 2^{31}$. These three integers represent that there is a direct road from intersection i to intersection j of length k. Roads can be traveled in either direction. There is at most one road between any two intersections.

Output

Print the minimum total distance you need to travel to reach home from school without getting stuck due to a flat tire. If the trip is not possible, output the word `stuck` on a single line instead.

It is guaranteed that if the trip is possible, the minimum distance D satisfies $0 < D < 2^{31}$.

Sample Explanation

In the first sample input, if your tire did not have a leak then the shortest path home would have a distance of 9, going from the school through intersections 3 and 5. However, due to the leak, you can only travel a distance of 4 before you need to refill the tire, requiring you to use the repair stations at intersections 2 and 5, for a total distance of 12.

In the second sample input, if your tire did not have a leak, then the shortest path home would have a distance of 12. But since your tire only lasts for a distance of 10, there's no path where your bike tire will not go flat somewhere along the way. Even when using repair station at intersection 2, you get stuck before you can reach either your home or the repair station at intersection 5.

Sample Input 1	Sample Output 1
6 7 2 4 2 5 1 2 4 1 3 5 2 3 3 3 4 2 3 5 1 4 6 4 5 6 4	12

Sample Input 2	Sample Output 2
6 7 2 10 2 5 1 2 1 1 3 2 2 3 1 3 4 8 4 5 3 4 6 2 5 6 1	stuck

Problem J
Stop Counting!

Photo by Drew Rae

The Martingale casino is creating new games to lure in new gamblers who tire of the standard fare. Their latest invention is a fast-paced game of chance called *Stop Counting!*, where a single customer plays with a dealer who has a deck of cards. Each card has some integer value.

One by one, the dealer reveals the cards in the deck in order, and keeps track of the sum of the played cards and the number of cards shown. At some point before a card is dealt, the player can call "Stop Counting!" After this, the dealer continues displaying cards in order, but does not include them in the running sums. At some point after calling "Stop Counting!", and just before another card is dealt, the player can also call "Start Counting!" and the dealer then includes subsequent cards in the totals. The player can only call "Stop Counting!" and "Start Counting!" at most once each, and they must call "Stop Counting!" before they can call "Start Counting!". A card is "counted" if it is dealt before the player calls "Stop Counting!" or is dealt after the player calls "Start Counting!"

The payout of the game is then the average value of the counted cards. That is, it is the sum of the counted cards divided by the number of counted cards. If there are no counted cards, the payout is 0.

You have an 'in' with the dealer, and you know the full deck in order ahead of time. What is the maximum payout you can achieve?

Input

The first line of the input contains a single integer $1 \leq N \leq 1\,000\,000$, the number of cards in the deck.

The second line of input contains N space-separated integers, the values on the cards. The value of each card is in the range $[-10^9, 10^9]$. The cards are dealt in the same order they are given in the input.

Output

Output the largest attainable payout. The answer is considered correct if the absolute error is less than 10^{-6}, or the relative error is less than 10^{-9}.

Sample Explanation

In the first sample, by calling "Stop Counting!" before the -10 and "Start Counting!" before the final 10, we can achieve an average of 10.0 with the cards that are counted.

In the second sample, all values are negative, so the best strategy is to call "Stop Counting!" before the first card is dealt and call "Start Counting!" after the last card is dealt. Since no cards were counted, the average of the counted cards is 0.0.

Sample Input 1	Sample Output 1
5 10 10 -10 -4 10	10.000000000

Sample Input 2	Sample Output 2
4 -3 -1 -4 -1	0.000000000

Problem K
Summer Trip

Cartoon of items related to travel. Image from mohamed_hassan at pixabay.com

Leo has started a job in a travel agency. His first task is to organize a summer trip to an exotic overseas city. During the summer season, events of various types take place in the city: sports matches, concerts, beach parties, and many others. At any given time, there is exactly one event taking place. Events of any particular type may take place more than once during the season. The itinerary of events that Leo offers to his clients cannot be chosen arbitrarily; the company requires them to form a so-called "good itinerary." A good itinerary is a consecutive sequence of at least two events in the summer season, where the first and last events are of different types, and they are both unique among all event types during the sequence. For example, if the first event in a good itinerary is a beach party, none of the other events during the itinerary can also be a beach party. There are no other restrictions on the event types in the sequence of a good itinerary.

Before he starts organizing the trip, Leo wants to know the total number of good itineraries that are possible given a calendar of events that will take place over the summer season.

Input

The input consists of one line with a string describing the sequence of event types in the summer season. All characters are lowercase English letters (`'a'–'z'`), with different letters represent different types of events. Character i of the string encodes the ith event of the summer. There are no blanks or spaces in the string.

The length of the input string is at least 2 and at most 100 000 characters.

Output

Print the number of good itineraries that exist for the given summer season.

Sample Input 1	Sample Output 1
abbcccddddeeeee	10

Sample Input 2	Sample Output 2
thenumberofgoodstringsis	143

This page is intentionally left blank.

Problem L
Traveling Merchant

Lithograph by Louis Haghe from an original by David Roberts, Wikipedia

There is a long east-west road which has n towns situated along it, numbered 1 to n from west to east. All towns buy and sell the same kind of goodie. The value of a goodie fluctuates according to a weekly schedule. A town buys and sells a goodie at its value in that town on that particular day. At town i, the value of a goodie changes by d_i every day in the first half of a week, and changes by $-d_i$ every day in the second half of a week. In other words, the value of a goodie at town i is v_i on Mondays and Sundays, $v_i + d_i$ on Tuesdays and Saturdays, $v_i + 2d_i$ on Wednesdays and Fridays, and $v_i + 3d_i$ on Thursdays.

A merchant is making a business travel plan. His trip begins at a starting town s and ends at a destination town t, visiting each town from s to t (inclusive) exactly once. The merchant starts the trip on a Monday. It takes him exactly one day to travel between adjacent towns and every day he travels to the next town on the path to the destination. He may buy exactly one goodie at a town along the trip and sell that goodie at a town he visits later. He can only buy once and sell once. The merchant would like to know the maximum possible profit of q travel plans with different choices of town s and town t.

Input

The first line of the input has a single integer n ($2 \le n \le 10^5$). The next n lines each have two integers. The ith line has v_i ($1 \le v_i \le 10^9$) and d_i ($1 \le v_i + 3d_i \le 10^9$). The next line has a single integer q ($1 \le q \le 10^5$). The following q lines each give a pair of integers s and t ($1 \le s, t \le n, s \ne t$), representing one travel plan from town s to town t. If $s < t$ the merchant travels west to east, otherwise he travels east to west.

Output

For each travel plan, output the maximum profit the merchant can make on a single line. If the merchant cannot make any profit, output 0.

Sample Input 1	Sample Output 1
5 1 2 2 1 5 0 4 -1 7 -2 5 1 5 5 1 3 1 4 5 5 4	4 2 2 1 0

Problem M
Zipline

Photo by Virginia State Parks.

A zipline is a very fun and fast method of travel. It uses a very strong steel cable, connected to two poles. A rider (which could be a person or some cargo) attaches to a pulley which travels on the cable. Starting from a high point on the cable, gravity pulls the rider along the cable.

Your friend has started a company which designs and installs ziplines, both for fun and for utility. However, there's one key problem: determining how long the cable should be between the two connection points. The cable must be long enough to reach between the two poles, but short enough that the rider is guaranteed to stay a safe distance above the ground. Help your friend determine these bounds on the length.

The cable connects to two vertical poles that are w meters apart, at heights g and h meters, respectively. You may assume that the cable is inelastic and has negligible weight compared to the rider, so that there is no sag or slack in the cable. That is, at all times the cable forms two straight line segments connecting the rider to the two poles, with the sum of the segments lengths equal to the total length of the cable. The lowest part of the rider hangs r meters below the cable; therefore the cable must stay at least r meters above the ground at all times during the ride. The ground is flat between the two poles. Please refer to the diagram in Figure M.1 for more information.

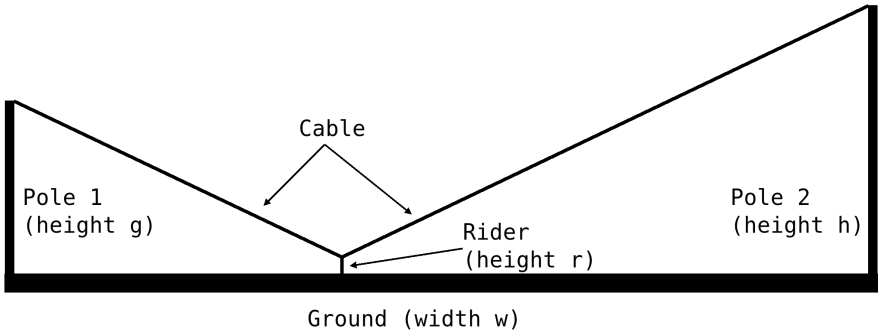

Figure M.1: A zipline, annotated with the four variables used to describe it.

Input

The input starts with a line containing an integer n, where $1 \leq n \leq 1\,000$. The next n lines each describe a zipline configuration with four integers: w, g, h, and r. These correspond to the variables described above. The limits on their values are: $1 \leq w, g, h \leq 1\,000\,000$, and $1 \leq r \leq \min(g, h)$.

Output

For each zipline, print a line of output with two lengths (in meters): the minimum and maximum length the cable can be while obeying the above constraints. Both lengths should have an absolute error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2 1000 100 100 20 100 20 30 2	1000.00000000 1012.71911209 100.49875621 110.07270325