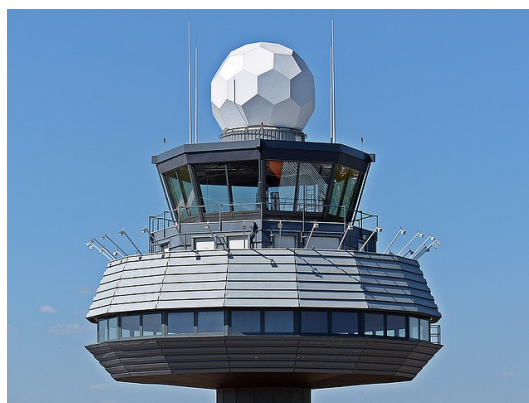


Air Traffic Control (atc)

Edoardo has recently been hired by the municipality of Milan as an Air Traffic Control operator. For his first week, he has been assigned by his boss to tackle the *paper-plane project*.

The paper-plane project is an effort to connect the city of Milan with a network of paper planes. To do this, Edoardo should build a number of control towers.



We can assume the city is represented by a square grid of size $N \times N$, where each cell can be either:

- Empty space, indicated with a “.” in input.
- An obstacle such as a mountain or building or other obstructing item, indicated with a “#” in input.
- A control tower, indicated with a “T” in input.

A control tower placed on cell (i, j) can throw a paper plane in four directions: up, down, left, right. The paper plane will travel in that direction until either another control tower or an obstacle is encountered. If a paper plane reaches a control tower, then it can be launched again from there.

The goal of this project is to connect the $(1, 1)$ tower to the (N, N) tower, which will always be present, so that it's possible to exchange paper planes between them. Help Edoardo compute the *minimum* number of towers he should construct in order to reach the goal. If such task is impossible regardless the number of additional towers built, then the whole paper-plane project is impossible.

Among the attachments of this task you may find a template file `atc.*` with a sample incomplete implementation.

Input

The first line contains an integer N , the size of the grid. The next N lines contain N characters each. The first character of the first line indicates the position $(1, 1)$ of the grid. The last character of the last line indicates the position (N, N) .

Output






You need to write a single line with an integer: the minimum number of towers that should be built to reach the goal. If it is impossible to reach the goal, write `-1`.

Constraints

- $2 \leq N \leq 200$.
- It is guaranteed that coordinates $(1, 1)$ and (N, N) always contain control towers.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N = 2$.

- **Subtask 3** (20 points) $N = 3$.

- **Subtask 4** (50 points) $N \leq 6$ and no more than 4 additional control towers are needed.

- **Subtask 5** (20 points) No additional limitations.


Examples

input	output
5 T # . . # . . # T	2
5 T . . . # . . . # . . . # . . . # . . . # . . . T	-1

Explanation

In the **first sample case**, it is enough to add 2 towers. For example, a possible solution is to add the towers highlighted in red:

```
T . . T .  
. . . . #  
. . # . .  
# . . . .  
. . . T T
```

In the **second sample case** it is impossible to connect towers at $(1, 1)$ and (N, N) .

Lovely Cats (cats)

Edoardo has finally built the new OIS building and he is ready to have a grand opening with a party. Being a lover of cats, Edoardo foresees that the small talk among his friends will revolve entirely on how cute these pets are.

He sees an opportunity: knowing that not everyone loves cats in the same manner, he might pair friends with different opinions on the subject so that who loves them most can try to persuade the other person.

This is not going to be easy. Edoardo, with a pragmatic approach, makes his friends line up along two lines: the left one with N_m male friends and the right one with N_f female friends. Then, he asks each one to express with a grade between 0 and 100 how much they love cats (M_i and F_i represent, respectively, the vote of the i -th male and of the i -th female in their lines).

Now Edoardo excludes some friends, getting them out of their lines, in order to reach a situation where the two lines (M' and F') are both of N friends and the quantity

$$Q = |M'_0 - F'_0| + |M'_1 - F'_1| + \dots + |M'_{N-1} - F'_{N-1}|$$

is the *maximum* possible. What is the maximum value of Q that he can reach?

Among the attachments of this task you may find a template file `cats.*` with a sample incomplete implementation.



Figure 1: A pretty cute cat. Lovely!

Input

The first line contains integers N_m and N_f . The second line contains N_m integers M_i . The third line contains N_f integers F_i .

Output






You need to write a single line with an integer: the maximum Q that Edoardo can obtain.

Constraints

- $1 \leq N_m, N_f \leq 1000$.
- $0 \leq M_i \leq 100$ for each $i = 0 \dots N_m - 1$.
- $0 \leq F_i \leq 100$ for each $i = 0 \dots N_f - 1$.
- Edoardo never changes the position of his friends in their lines. When the i -th is excluded from a line, every friend in that line at position $i + 1, i + 2, \dots$ moves one position left. It is never allowed to leave a “hole” in the line.
- Edoardo can exclude any number (0 included) of males and females at his sole discretion to maximize Q .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (20 points) Every male friend expressed the same vote: $M_0 = M_1 = \dots = M_{N_m-1}$.

- **Subtask 3** (20 points) Votes are non-decreasing: $M_i \leq M_{i+1}$ for each $i = 0 \dots N_m - 2$ and $F_i \leq F_{i+1}$ for each $i = 0 \dots N_f - 2$.

- **Subtask 4** (25 points) $N_m, N_f \leq 10$.

- **Subtask 5** (35 points) No additional limitations.


Examples

input	output
3 3 0 10 100 50 70 80	140
3 4 0 10 10 100 0 100 100	280

Explanation

In the **first sample case**, if Edoardo leaves the friends as they already are, he gets $Q = 50 + 60 + 20 = 130$. There is a better move: to exclude the last male and the first female to get $Q = 70 + 70 = 140$. There is no better strategy.

In the **second sample case** the best strategy is to exclude the second female to obtain $Q = 100 + 90 + 90 = 280$.

Unreadable Dates (dates)

A friend of Luca is particularly interested in history: he is writing his final thesis to graduate in this subject. During his researches, he is writing an appendix listing all the dates of death of the soldiers who fought in a certain battle.

He has already found an old book which is perfectly suited to accomplish the task, but he has been unlucky: he is unable to read from the yellowed pages the separator (‘/’) between days, months and years of the dates.

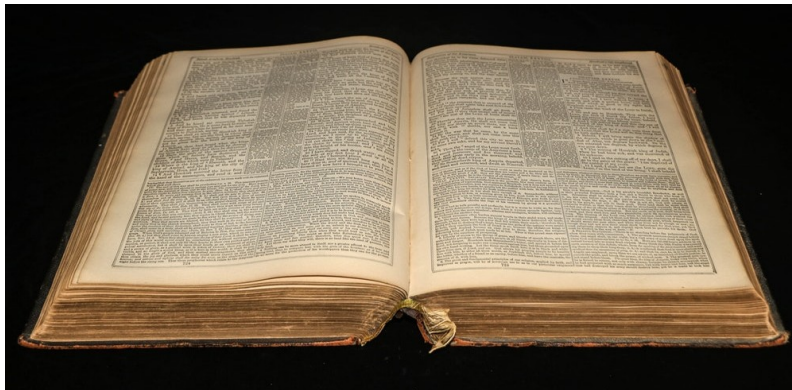


Figure 1: The old book which *almost* perfectly satisfies Luca’s friend.

You know that:

- The day is a number between 1 and 31 (except for shorter months), optionally preceded by a zero when it is a single digit (i.e., 05 and 5 are both possible).
- The month is a number between 1 and 12, optionally preceded by a zero when it is a single digit. All months have 31 days except April, June, September and November which have 30 days and February which has 28 days.
- The year is between 1900 and 1999 and may be written with two digits (the last two), three digits (the last three) or with all four digits.

Help Luca’s friend, who is unsure how to proceed. Given a string `date`, exactly as read from the book without separators, how many possible different valid dates may be interpreted from it?

Among the attachments of this task you may find a template file `dates.*` with a sample incomplete implementation.

Input

The first and only line contains the string of digits `date`.

Output






You need to write a single line with an integer: the number of different dates in which the input string can be interpreted.

Constraints

- `date` has a length between 4 and 8 digits (extremes included).
- We are not interested in considering leap years: February has always 28 days.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (25 points) `date` is always 8 digits long and can represent only dates which have a month with 31 days.

- **Subtask 3** (15 points) `date` is always 8 digits long.

- **Subtask 4** (15 points) `date` can represent only dates which have always both the day and the month written with two digits each.

- **Subtask 5** (40 points) No additional limitations.


Examples

input	output
10119	2
21121999	1
311399	0

Explanation

In the **first sample case** the date can be interpreted as 10/1/19 or as 1/1/19 (with a leading zero in front of the month).

In the **second sample case** the date can be interpreted only as 21/12/1999.

In the **third sample case** there is no valid date which can be deduced from the input string.

Text Editor (editor)

Giorgio was asked to write the minutes of the last round of the *Olimpiadi di Informatica a Squadre*. After writing thousands of words in his beloved *MS Notepad*¹, he now wants to format them properly and send the result via e-mail to the organizers. Nothing easier right? Except his e-mail client supports only formatting with **bold** and *italic*! At least you can mix them as much as you want, for example **this sentence is a mix of both** the formatters.

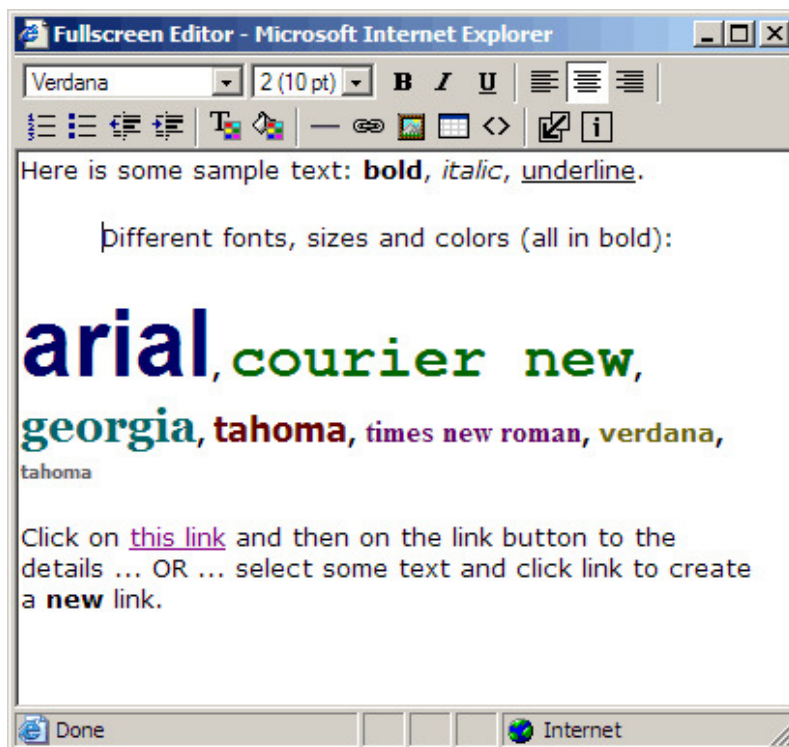


Figure 1: Is there something Internet Explorer can't do?

In Giorgio's editor, in order to toggle the bold-ness or italic-ness of some text you can simply select a continuous range of characters with your mouse and then press the button **toggle bold** or the button *toggle italic*. By doing so all the not-bold characters will become bold and vice-versa, and respectively for the italic ones.

Giorgio has already chosen which parts should be italic and which bold (and which both): your task is to tell him how many times he has to press the toggle buttons at minimum.

👉 Among the attachments of this task you may find a template file `editor.*` with a sample incomplete implementation.

¹Mega Software™

Input

The first line contains the only integer N . The second line contains N characters V_i separated by spaces. Each character is among 'n', 'i', 'N', 'I'. If the i -th character is letter i ('i' or 'I'), then the corresponding character in the text should be *italic*. If the i -th character is uppercase ('N' or 'I'), then the corresponding character in the text should be **bold**.

Output







You need to write a single line with an integer: the minimum number of buttons to press.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- V_i is among 'n', 'i', 'N', 'I'.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N \leq 10$.

- **Subtask 3** (15 points) There is no bold character: V_i is among 'n' or 'i'.

- **Subtask 4** (15 points) There is no italic character: V_i is among 'n' or 'N'.

- **Subtask 5** (30 points) $N \leq 1000$.

- **Subtask 6** (30 points) No additional limitations.


Examples

input	output
6 n N I I N n	2
9 N I I i n n i i N	4

Explanation

Counting from 0, in the **first sample case** you can select characters 1..4 and then use **toggle bold**, then select 2..3 and then use *toggle italic*.

In the **second sample case**, you can toggle **bold** characters 0..2, *italic* 1..7, *italic* 4..5 and **bold** 8.

Art Gallery Selection (gallery)

Luca decided to enter the art business, and is opening a new modern art gallery in Milan. As soon as he opened a call for young emerging artists, his e-mail was flooded by pictures of paintings! In fact, too much to be selected by hand: he needs an algorithm to extract the most promising among them. Following blindly his profound artistic sense, developed in years studying engineering, Luca believes that the most beautiful paintings are those featuring subjects arranged symmetrically with respect to a vertical axis.



Figure 1: Few of the paintings that Luca has to sort.

Given a description of a painting, picturing N main subjects, each of them on a specific coordinate (X_i, Y_i) , help Luca determine whether there exists a vertical axis of symmetry for the given points.

👉 Among the attachments of this task you may find a template file `gallery.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The following N lines contains two integers X_i, Y_i .

Output

You need to write a single line with the string 'YES' if a vertical axis of symmetry exists, or the string 'NO' if it does not.

Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq X_i, Y_i \leq 100\,000$ for each $i = 0 \dots N - 1$.
- The points are all distinct.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

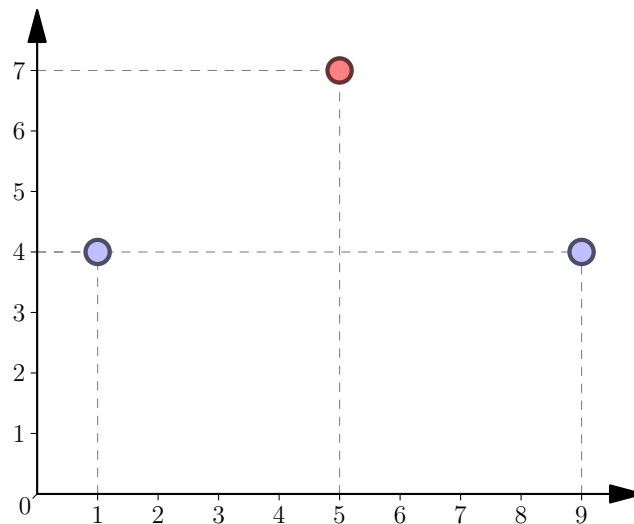
- **Subtask 1** (0 points) Examples.
🟡🟡🟡🟡🟡
- **Subtask 2** (10 points) $N \leq 50$ and Y_i are all equal.
🟡🟡🟡🟡🟡
- **Subtask 3** (30 points) $N \leq 200$ and Y_i are all equal.
🟡🟡🟡🟡🟡
- **Subtask 4** (25 points) $N \leq 3000$ and Y_i are all equal.
🟡🟡🟡🟡🟡
- **Subtask 5** (15 points) $N \leq 3000$.
🟡🟡🟡🟡🟡
- **Subtask 6** (20 points) No additional limitations.
🟡🟡🟡🟡🟡

Examples

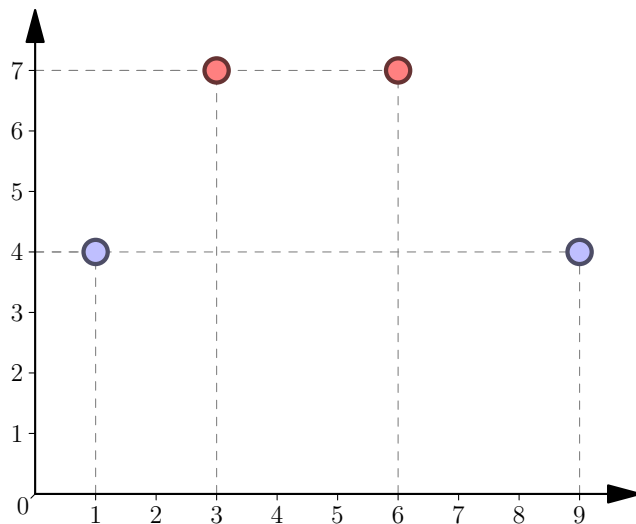
input	output
3 5 7 1 4 9 4	YES
4 1 4 9 4 3 7 6 7	NO

Explanation

In the **first sample case**, a vertical symmetry axis exists at $X = 5$.



In the **second sample case**, two points would be symmetric for $X = 5$, and other two for $X = 4.5$: overall, no vertical symmetry axis exists.



The Kabbalah Returns (kabbalah2)

Luca heard from George about the powers of the *Kabbalah* and now he wants to try for himself. He promptly enrolled in the same course as George, where students routinely spend days and nights studying the wisdoms hidden in the Bible.

Luca learned that in order to become the master of all masters it is crucial to list all the numbers contained in all pages of the Bible, and put those numbers in a $N \times M$ grid. Doing so will make the numbers entangle together and release the hidden power!



After preparing the matrix, Luca needs to find a clue: the year of the edition of the *master book*, a particular Bible which is thought to be the most powerful one. The year is defined as the 4-digit number (without leading zeros) that appears most frequently in the grid. When looking for these occurrences, all sequences of 4 adjacent positions in the 8 cardinal directions are considered (see examples).

👉 Among the attachments of this task you may find a template file `kabbalah2.*` with a sample incomplete implementation.

Input

The first line contains 2 integers N and M , the number of rows and the number of columns. The next N lines contain M characters long numbers.

Output







You need to write a single line with two integers: the four digit number with the highest number of occurrences (without leading zeros) and the number of occurrences of that number in the grid.

Constraints

- $4 \leq N, M \leq 1000$.
- Each digit in the grid is between 0 and 9.
- If there are multiple solutions, choose the smallest one (the smallest edition year).

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

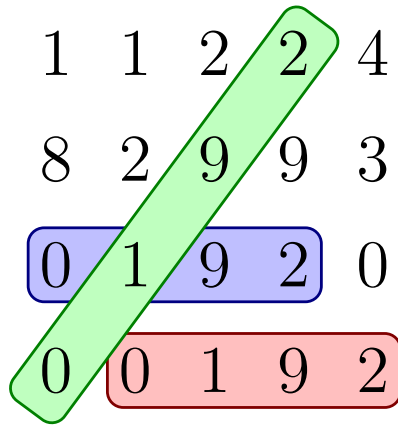
- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N = M = 4$ and all the digits are equal.

- **Subtask 3** (10 points) All the digits are equal.

- **Subtask 4** (15 points) All the digits are zeros except exactly one.

- **Subtask 5** (30 points) $N \leq 100$.

- **Subtask 6** (35 points) No additional limitations.


Examples

input	output
4 5 11224 82993 01920 00192	2910 3
10 10 8747832883 2423768395 6346456432 5324532466 3246326365 2564564634 3456346534 4563464365 6435634456 4536346545	6666 8

Explanation

In the **first sample case** the searched numbers are highlighted in the following picture.



In the **second sample case** there are 8 occurrences of 6666, note that many of them overlap and are counted twice because 6666 can be read also from right to left.

Persian Party (mehmooni)

William is a regular participant at Iranian parties. In Persian, a party is called **mehmooni** and is quite different from normal parties: it is usually thrown in a private house, where the host (the owner of the house) will invite N guests and try to stuff them with *tahdig* and *kabab*. It's no wonder some people say that “a mehmooni is the fastest way to end up in a hospital or gain 10kg”.



Figure 1: A typical Mehmooni.

What makes a mehmooni different than other parties, though, is a custom that is typical of the (overly polite?) Iranian culture: every time some new guest arrives at the party, he or she will go to shake hands and say *salam* (“hello”) to the host and to **every other guest who is present at that time**, one by one. The same goes for when a guest leaves the party: they will again go shake hands and say *khodahafez* (“goodbye”) to the host and to every guest present at that time. Note that the guests who are present at those two different moments could be different.

Even though William is not Iranian, he respects the tradition and always obeys the hand shaking rule. However, he is wondering just **how many handshakes** will be exchanged in a mehmooni, assuming to be able to know exactly when each guest will *arrive* to the party and when they will *leave* it.

Among the attachments of this task you may find a template file `mehmooni.*` with a sample incomplete implementation.

Input

The first line contains an integer N , the number of guests invited to the mehmooni. The next N lines contain two integers each: A_i and D_i , respectively the *arrival* and *departure* time of the i -th guest. The time is measured in seconds elapsed since the start of the mehmooni.

Output





You need to write a single line with an integer: the total number of handshakes that will happen.

Constraints

- $1 \leq N \leq 200\,000$.
- $1 \leq A_i < D_i \leq 10^9$ for each $i = 0 \dots N - 1$.
- No two guests will arrive or leave at the same exact moment: the door is not large enough!
- For the same reason, if some guest is arriving at time t then no guest can leave exactly at time t .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (30 points) $N \leq 100$.

- **Subtask 3** (50 points) $N \leq 1000$.

- **Subtask 4** (20 points) No additional limitations.


Examples

input	output
4 6 8 2 4 1 5 3 7	16
4 7 8 3 4 5 6 1 2	8

Explanation

In the **first sample case** we can take the first guest as an example: the guest is one of the last people to arrive ($t = 6$) and at that time some of the other guests already left so there are just 2 people to shake hands with: one guest and the host of the party. That guest is also the last person to leave: at that time there is just the host to shake hands with.

In the **second sample case** the mehmooni is especially boring: there is always at most 1 guest in the house at any moment, thus they only shake hands with the host (twice, while arriving and while departing).

Compulsive Shopping (shopping)

William's girlfriend is a compulsive shopper. Every time she notices a nice item while on a walk with her significant other, she desperately wants to have it! It goes without saying that William should pay for it, as a true gentleman... and this habit is wreaking havoc on his precarious finances, now limited to merely E euros total.




Figure 1: William's girlfriend after noticing a must-have pair of shoes.

In order to avoid bankruptcy, William went for a patrol before the next walk, and listed N items of price P_i that his girlfriend will want to have (in appearance order). William knows very well that he can refuse to buy an item desired by his beloved in two cases only:

1. if he cannot afford it, as it costs more than his current total finances;
2. if he bought the last item encountered.

Refusing to buy an item in any other case would result in an prompt breakup, that is not an option. Help William plan how much euros to waste before the walk and then which items to buy, in order to complete the walk with as much remaining euros as possible, without breaking up with his girlfriend!

 Among the attachments of this task you may find a template file `shopping.*` with a sample incomplete implementation.

Input

The first line contains two integers N , E . The second line contains N integers P_i .

Output









You need to write a single line with an integer: the maximum amount of euros William can save without breaking up with his girlfriend.

Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq E \leq 10^9$.
- $0 \leq P_i \leq 1\,000\,000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $P_i = P_j$ for all i, j .

- **Subtask 3** (15 points) $E = 10^9, P_i \leq 1000$.

- **Subtask 4** (25 points) $N \leq 20, E \leq 2000$.

- **Subtask 5** (20 points) $N \leq 200, E \leq 2000$.

- **Subtask 6** (15 points) $N, E \leq 10\,000$.

- **Subtask 7** (5 points) $N \leq 10\,000$.

- **Subtask 8** (10 points) No additional limitations.


Examples

input	output
3 35 20 5 10	14
5 80 10 50 90 30 20	40

Explanation

In the **first sample case**, if William starts the walk with all of its 35€, he is able to afford everything so case (1) does not apply. By (2), he has to buy item 1 but then he can choose between items 2 and 3, the best being buying the first two items for a final $35 - 20 - 5 = 10\text{€}$. However, William can do better by wasting 16€ before the walk, thus starting with $35 - 16 = 19\text{€}$: in this case, he cannot afford the first item and buys the second ending with $19 - 5 = 14\text{€}$.

In the **second sample case**, William can buy the first and fourth item, ending with $80 - 10 - 30 = 40\text{€}$. In this case, wasting money before the walk does not help in improving the result.