VK Cup 2018 - Wild-card Round 1

A. 2-3-numbers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A positive integer is called a *2-3-integer*, if it is equal to $2^x \cdot 3^y$ for some non-negative integers x and y. In other words, these integers are such integers that only have 2 and 3 among their prime divisors. For example, integers 1, 6, 9, 16 and 108 — are 2-3 integers, while 5, 10, 21 and 120 are not.

Print the number of *2-3-integers* on the given segment $[l, r]$, i. e. the number of sich *2-3-integers* t that $l \le t \le r$.

Input
The only line contains two integers l and r $(1 \le l \le r \le 2 \cdot 10^9)$.

Output
Print a single integer the number of *2-3-integers* on the segment $[l, r]$.

Examples

input
1 10
output
7

input
100 200
output
5

input
1 2000000000
output
326

Note
In the first example the *2-3-integers* are 1, 2, 3, 4, 6, 8 and 9.

In the second example the *2-3-integers* are 108, 128, 144, 162 and 192.

B. Add Points

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n points on a straight line, and the i-th point among them is located at x_i. All these coordinates are distinct.

Determine the number m — the smallest number of points you should add on the line to make the distances between all neighboring points equal.

Input
The first line contains a single integer n $(3 \le n \le 100\,000)$ — the number of points.

The second line contains a sequence of integers $x_1, x_2, \ldots, x_n$ ($-10^9 \le x_i \le 10^9$) — the coordinates of the points. All these coordinates are distinct. The points can be given in an arbitrary order.

Output
Print a single integer m — the smallest number of points you should add on the line to make the distances between all neighboring points equal.

Examples

input

```
3
-5 10 5
```

output

```
1
```

input

```
6
100 200 400 300 600 500
```

output

```
0
```

input

```
4
10 9 0 -1
```

output

```
8
```

Note

In the first example you can add one point with coordinate 0.

In the second example the distances between all neighboring points are already equal, so you shouldn't add anything.

C. Is This a Zebra?

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A camera you have accidentally left in a desert has taken an interesting photo. The photo has a resolution of n pixels width, and each column of this photo is all white or all black. Thus, we can represent the photo as a sequence of n zeros and ones, where 0 means that the corresponding column is all white, and 1 means that the corresponding column is black.

You think that this photo can contain a zebra. In this case the whole photo should consist of several (possibly, only one) alternating black and white stripes of equal width. For example, the photo $[0, 0, 0, 1, 1, 1, 0, 0, 0]$ can be a photo of zebra, while the photo $[0, 0, 0, 1, 1, 1, 1]$ can not, because the width of the black stripe is 3, while the width of the white stripe is 4. Can the given photo be a photo of zebra or not?

Input

The first line contains a single integer n $(1 \le n \le 100\,000)$ — the width of the photo.

The second line contains a sequence of integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 1)$ — the description of the photo. If a_i is zero, the i-th column is all black. If a_i is one, then the i-th column is all white.

Output

If the photo can be a photo of zebra, print "YES" (without quotes). Otherwise, print "NO".

You can print each letter in any case (upper or lower).

Examples

input

```
9
0 0 0 1 1 1 0 0 0
```

output

```
YES
```

input

```
7
0 0 0 1 1 1 1
```

output

```
NO
```

input

```
5
1 1 1 1 1
```

output

```
YES
```

input

```
8
1 1 1 0 0 0 1 1
```

output

```
NO
```

input

```
9
1 1 0 1 1 0 1 1 0
```

output

```
NO
```

Note

The first two examples are described in the statements.

In the third example all pixels are white, so the photo can be a photo of zebra.

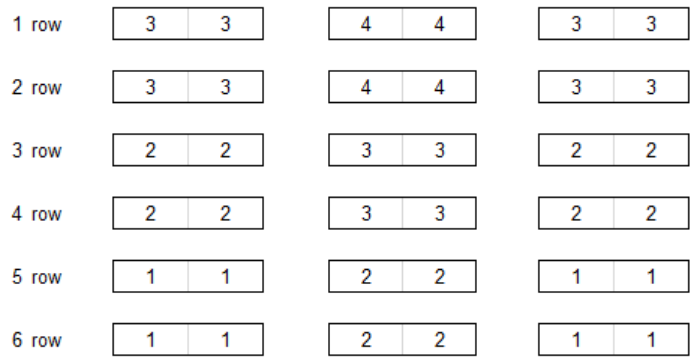
In the fourth example the width of the first stripe is equal to three (white color), the width of the second stripe is equal to three (black), and the width of the third stripe is equal to two (white). Thus, not all stripes have equal length, so this photo is not a photo of zebra.

D. Choose Place

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A classroom in a school has six rows with 3 desks in each row. Two people can use the same desk: one sitting on the left and one sitting on the right.

Some places are already occupied, and some places are vacant. Petya has just entered the class and wants to occupy the most convenient place. The conveniences of the places are shown on the picture:



Here, the desks in the top row are the closest to the blackboard, while the desks in the bottom row are the furthest from the blackboard.

You are given a plan of the class, where '`*`' denotes an occupied place, '`.`' denotes a vacant place, and the aisles are denoted by '`-`'.

Find any of the most convenient vacant places for Petya.

Input

The input consists of 6 lines. Each line describes one row of desks, starting from the closest to the blackboard. Each line is given in the following format: two characters, each is '`*`' or '`.`' — the description of the left desk in the current row; a character '`-`' — the aisle; two characters, each is '`*`' or '`.`' — the description of the center desk in the current row; a character '`-`' — the aisle; two characters, each is '`*`' or '`.`' — the description of the right desk in the current row. So, the length of each of the six lines is 8.

It is guaranteed that there is at least one vacant place in the classroom.

Output

Print the plan of the classroom after Petya takes one of the most convenient for him places. Mark this place with the letter '`P`'. There should be exactly one letter '`P`' in the plan. Petya can only take a vacant place. In all other places the output should coincide with the input.

If there are multiple answers, print any.

Examples

input

```
..-**-..
..-**-..
..-..-..
..-..-..
..-..-..
..-..-..
```

output

```
..-**-..
..-**-..
..-..-..
..-P.-..
..-..-..
..-..-..
```

input

```
**-**-**
**-**-**
..-**-.*
**-**-**
..-..-..
..-**-..
```

output

```
**-**-**
**-**-**
..-**-.*
**-**-**
..-P.-..
..-**-..
```

input

```
**-**-*.
*.-*.-**
**-**-**
**-**-**
..-..-..
..-**-..
```

output

```
**-**-*.
*.-*P-**
**-**-**
**-**-**
..-..-..
..-**-..
```

Note

In the first example the maximum convenience is 3.

In the second example the maximum convenience is 2.

In the third example the maximum convenience is 4.

E. Merge Equal Elements

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a sequence of positive integers $a_1, a_2, \ldots, a_n$.

While possible, you perform the following operation: find a pair of equal consecutive elements. If there are more than one such pair, find the leftmost (with the smallest indices of elements). If the two integers are equal to x, delete both and insert a single integer $x + 1$ on their place. This way the number of elements in the sequence is decreased by 1 on each step.

You stop performing the operation when there is no pair of equal consecutive elements.

For example, if the initial sequence is $[5, 2, 1, 1, 2, 2]$, then after the first operation you get $[5, 2, 2, 2, 2]$, after the second — $[5, 3, 2, 2]$, after the third — $[5, 3, 3]$, and finally after the fourth you get $[5, 4]$. After that there are no equal consecutive elements left in the sequence, so you stop the process.

Determine the final sequence after you stop performing the operation.

Input

The first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of elements in the sequence.

The second line contains the sequence of integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output

In the first line print a single integer k — the number of elements in the sequence after you stop performing the operation.

In the second line print k integers — the sequence after you stop performing the operation.

Examples

input

```
6
5 2 1 1 2 2
```

output

```
2
5 4
```

input

```
4
1000000000 1000000000 1000000000 1000000000
```

output

```
1
1000000002
```

input

```
7
4 10 22 11 12 5 6
```

output

```
7
4 10 22 11 12 5 6
```

Note

The first example is described in the statements.

In the second example the initial sequence is $[1000000000, 1000000000, 1000000000, 1000000000]$. After the first operation the sequence is equal to $[1000000001, 1000000000, 1000000000]$. After the second operation the sequence is $[1000000001, 1000000001]$. After the third operation the sequence is $[1000000002]$.

In the third example there are no two equal consecutive elements initially, so the sequence does not change.

F. Mobile Communications

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A sum of p rubles is charged from Arkady's mobile phone account every day in the morning. Among the following m days, there are n days when Arkady will top up the account: in the day d_i he will deposit t_i rubles on his mobile phone account. Arkady will always top up the account before the daily payment will be done. There will be no other payments nor tops up in the following m days.

Determine the number of days starting from the 1-st to the m-th such that the account will have a negative amount on it after the daily payment (i. e. in evening). Initially the account's balance is zero rubles.

Input

The first line contains three integers n, p and m ($1 \le n \le 100\,000$, $1 \le p \le 10^9$, $1 \le m \le 10^9$, $n \le m$) — the number of days Arkady will top up the account, the amount of the daily payment, and the number of days you should check.

The i-th of the following n lines contains two integers d_i and t_i ($1 \le d_i \le m$, $1 \le t_i \le 10^9$) — the index of the day when Arkady will make the i-th top up, and the amount he will deposit on this day. It is guaranteed that the indices of the days are distinct and are given in increasing order, i. e. $d_i > d_{i-1}$ for all i from 2 to n.

Output

Print the number of days from the 1-st to the m-th such that the account will have a negative amount on it after the daily payment.

Examples

input

```
3 6 7
2 13
4 20
7 9
```

output

```
3
```

input

```
5 4 100
10 70
15 76
21 12
30 100
67 85
```

output

```
26
```

Note

In the first example the balance will change as following (remember, initially the balance is zero):

1. in the first day 6 rubles will be charged, the balance in the evening will be equal to -6;
2. in the second day Arkady will deposit 13 rubles, then 6 rubles will be charged, the balance in the evening will be equal to 1;
3. in the third day 6 rubles will be charged, the balance in the evening will be equal to -5;
4. in the fourth day Arkady will deposit 20 rubles, then 6 rubles will be charged, the balance in the evening will be equal to 9;
5. in the fifth day 6 rubles will be charged, the balance in the evening will be equal to 3;
6. in the sixth day 6 rubles will be charged, the balance in the evening will be equal to -3;
7. in the seventh day Arkady will deposit 9 rubles, then 6 rubles will be charged, the balance in the evening will be equal to 0.

Thus, in the end of the first, third and sixth days the balance will be negative in the end of the day.

G. Large Bouquets

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A flower shop has got n bouquets, and the i-th bouquet consists of a_i flowers. Vasya, the manager of the shop, decided to make large bouquets from these bouquets.

Vasya thinks that a bouquet is large if it is made of **two or more** initial bouquets, and there is a constraint: the total number of flowers in a large bouquet should be **odd**. Each of the initial bouquets can be a part of at most one large bouquet. If an initial bouquet becomes a part of a large bouquet, all its flowers are included in the large bouquet.

Determine the maximum possible number of large bouquets Vasya can make.

Input

The first line contains a single positive integer n ($1 \le n \le 10^5$) — the number of initial bouquets.

The second line contains a sequence of integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the number of flowers in each of the initial bouquets.

Output

Print the maximum number of large bouquets Vasya can make.

Examples

input
5 2 3 4 2 7
output
2

input
6 2 2 6 8 6 12
output
0

input
3 11 4 10
output
1

Note

In the first example Vasya can make 2 large bouquets. For example, the first bouquet can contain the first and the fifth initial bouquets (the total number of flowers is then equal to 9), and the second bouquet can consist of the second and the third initial bouquets (the total number of flowers is then equal to 7). The fourth initial bouquet is unused in this scheme.

In the second example it is not possible to form a single bouquet with odd number of flowers.

In the third example Vasya can make one large bouquet. For example, he can make it using all three initial bouquets. The size of the large bouquet is then equal to $11 + 4 + 10 = 25$.

H. Endless Roses Most Beautiful

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Arkady decided to buy roses for his girlfriend.

A flower shop has white, orange and red roses, and the total amount of them is n. Arkady thinks that red roses are not good together with white roses, so he won't buy a bouquet containing both red and white roses. Also, Arkady won't buy a bouquet where all roses have the same color.

Arkady wants to buy exactly k roses. For each rose in the shop he knows its beauty and color: the beauty of the i-th rose is b_i, and its color is c_i ('W' for a white rose, 'O' for an orange rose and 'R' for a red rose).

Compute the maximum possible total beauty of a bouquet of k roses satisfying the constraints above or determine that it is not possible to make such a bouquet.

Input

The first line contains two integers n and k ($1 \le k \le n \le 200\,000$) — the number of roses in the show and the number of roses Arkady wants to buy.

The second line contains a sequence of integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10\,000$), where b_i equals the beauty of the i-th rose.

The third line contains a string c of length n, consisting of uppercase English letters 'W', 'O' and 'R', where c_i denotes the color of the i-th rose: 'W' denotes white, 'O' — orange, 'R' — red.

Output

Print the maximum possible total beauty of a bouquet of k roses that satisfies the constraints above. If it is not possible to make a single such bouquet, print -1.

Examples

input
5 3 4 3 4 1 6 RROWW
output
11

input
5 2 10 20 14 20 11 RRRRR
output
-1

input
11 5 5 6 3 2 3 4 7 5 4 5 6 RWOORWORROW
output
28

Note

In the first example Arkady wants to buy 3 roses. He can, for example, buy both red roses (their indices are 1 and 2, and their total beauty is 7) and the only orange rose (its index is 3, its beauty is 4). This way the total beauty of the bouquet is 11.

In the second example Arkady can not buy a bouquet because all roses have the same color.

I. A Vital Problem

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a strict daily schedule. He has n alarms set for each day, and the i-th alarm rings each day at the same time during exactly one minute.

Determine the longest time segment when Polycarp can sleep, i. e. no alarm rings in that period. It is possible that Polycarp begins to sleep in one day, and wakes up in another.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of alarms.

Each of the next n lines contains a description of one alarm. Each description has a format "hh:mm", where hh is the hour when the alarm rings, and mm is the minute of that hour when the alarm rings. The number of hours is between 0 and 23, and the number of minutes is between 0 and 59. All alarm times are distinct. The order of the alarms is arbitrary.

Each alarm starts ringing in the beginning of the corresponding minute and rings for exactly one minute (i. e. stops ringing in the beginning of the next minute). Polycarp can start sleeping instantly when no alarm is ringing, and he wakes up at the moment when some alarm starts ringing.

Output

Print a line in format "hh:mm", denoting the maximum time Polycarp can sleep continuously. hh denotes the number of hours, and mm denotes the number of minutes. The number of minutes should be between 0 and 59. Look through examples to understand the format better.

Examples

input
1 05:43
output
23:59

input
4 22:00 03:21 16:03 09:59
output
06:37

Note

In the first example there is only one alarm which rings during one minute of a day, and then rings again on the next day, 23 hours and 59 minutes later. Polycarp can sleep all this time.

J. Segments

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a straight line colored in white. n black segments are added on it one by one.

After each segment is added, determine the number of connected components of black segments (i. e. the number of black segments in the union of the black segments).

In particular, if one segment ends in a point x, and another segment starts in the point x, these two segments belong to the same connected component.

Input

The first line contains a single integer n ($1 \le n \le 200\,000$) — the number of segments.

The i-th of the next n lines contains two integers l_i and r_i ($1 \le l_i < r_i \le 10^9$) — the coordinates of the left and the right ends of the i-th segment. The segments are listed in the order they are added on the white line.

Output

Print n integers — the number of connected components of black segments after each segment is added.

Examples

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

input
9 10 20 50 60 30 40 70 80 90 100 60 70

```
10 40
40 50
80 90
```

output

```
1 2 3 4 5 4 3 2 1
```

Note

In the first example there are two components after the addition of the first two segments, because these segments do not intersect. The third added segment intersects the left segment and touches the right segment at the point 4 (these segments belong to the same component, according to the statements). Thus the number of connected components of black segments is equal to 1 after that.