Codeforces Round #393 (Div. 2) (8VC Venture Cup 2017 - Final Round Div. 2 Edition)

A. Petr and a calendar

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petr wants to make a calendar for current month. For this purpose he draws a table in which columns correspond to weeks (a week is seven consequent days from Monday to Sunday), rows correspond to weekdays, and cells contain dates. For example, a calendar for January 2017 should look like on the picture:

Petr wants to know how many columns his table should have given the month and the weekday of the first date of that month? Assume that the year is non-leap.

Input

The only line contain two integers m and d $(1 \leq m \leq 12,\ 1 \leq d \leq 7)$ — the number of month (January is the first month, December is the twelfth) and the weekday of the first date of this month (1 is Monday, 7 is Sunday).

Output

Print single integer: the number of columns the table should have.

Examples

input
```
1 7
```
output
```
6
```

input
```
1 1
```
output
```
5
```

input
```
11 6
```
output
```
5
```

Note

The first example corresponds to the January 2017 shown on the picture in the statements.

In the second example 1-st January is Monday, so the whole month fits into 5 columns.

In the third example 1-st November is Saturday and 5 columns is enough.

B. Frodo and pillows

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

n hobbits are planning to spend the night at Frodo's house. Frodo has n beds standing in a row and m pillows ($n \le m$). Each hobbit needs a bed and at least one pillow to sleep, however, everyone wants as many pillows as possible. Of course, it's not always possible to share pillows equally, but any hobbit gets hurt if he has at least two pillows less than some of his neighbors have.

Frodo will sleep on the k-th bed in the row. What is the maximum number of pillows he can have so that every hobbit has at least one pillow, every pillow is given to some hobbit and no one is hurt?

Input

The only line contain three integers n, m and k ($1 \le n \le m \le 10^9$, $1 \le k \le n$) — the number of hobbits, the number of pillows and the number of Frodo's bed.

Output

Print single integer — the maximum number of pillows Frodo can have so that no one is hurt.

Examples

input
```
4 6 2
```
output
```
2
```

input
```
3 10 3
```
output
```
4
```

input
```
3 6 1
```
output
```
3
```

Note

In the first example Frodo can have at most two pillows. In this case, he can give two pillows to the hobbit on the first bed, and one pillow to each of the hobbits on the third and the fourth beds.

In the second example Frodo can take at most four pillows, giving three pillows to each of the others.

In the third example Frodo can take three pillows, giving two pillows to the hobbit in the middle and one pillow to the hobbit on the third bed.

C. Pavel and barbecue

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Pavel cooks barbecue. There are n skewers, they lay on a brazier in a row, each on one of n positions. Pavel wants each skewer to be cooked some time in every of n positions in two directions: in the one it was directed originally and in the reversed direction.

Pavel has a plan: a permutation p and a sequence $b_1, b_2, ..., b_n$, consisting of zeros and ones. Each second Pavel move skewer on position i to position p_i, and if b_i equals 1 then he reverses it. So he hope that every skewer will visit every position in both directions.

Unfortunately, not every pair of permutation p and sequence b suits Pavel. What is the minimum total number of elements in the given permutation p and the given sequence b he needs to change so that every skewer will visit each of $2n$ placements? Note that after changing the permutation should remain a permutation as well.

There is no problem for Pavel, if some skewer visits some of the placements several times before he ends to cook. In other words, a permutation p and a sequence b suit him if there is an integer k ($k \geq 2n$), so that after k seconds each skewer visits each of the $2n$ placements.

It can be shown that some suitable pair of permutation p and sequence b exists for any n.

Input

The first line contain the integer n $(1 \leq n \leq 2 \cdot 10^5)$ — the number of skewers.

The second line contains a sequence of integers $p_1, p_2, ..., p_n$ $(1 \leq p_i \leq n)$ — the permutation, according to which Pavel wants to move the skewers.

The third line contains a sequence $b_1, b_2, ..., b_n$ consisting of zeros and ones, according to which Pavel wants to reverse the skewers.

Output

Print single integer — the minimum total number of elements in the given permutation p and the given sequence b he needs to change so that every skewer will visit each of $2n$ placements.

Examples

input

```
4
4 3 2 1
0 1 1 1
```

output

```
2
```

input

```
3
2 3 1
0 0 0
```

output

```
1
```

Note

In the first example Pavel can change the permutation to $4, 3, 1, 2$.

In the second example Pavel can change any element of b to 1.

D. Travel Card

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A new innovative ticketing systems for public transport is introduced in Bytesburg. Now there is a single travel card for all transport. To make a trip a passenger scan his card and then he is charged according to the fare.

The fare is constructed in the following manner. There are three types of tickets:

1. a ticket for one trip costs 20 byteland rubles,
2. a ticket for 90 minutes costs 50 byteland rubles,
3. a ticket for one day (1440 minutes) costs 120 byteland rubles.

Note that a ticket for x minutes activated at time t can be used for trips started in time range from t to $t + x - 1$, inclusive. Assume that all trips take exactly one minute.

To simplify the choice for the passenger, the system automatically chooses the optimal tickets. After each trip starts, the system analyses all the previous trips and the current trip and chooses a set of tickets for these trips with a minimum total cost. Let the minimum total cost of tickets to cover all trips from the first to the current is a, and the total sum charged before is b. Then the system charges the passenger the sum $a - b$.

You have to write a program that, for given trips made by a passenger, calculates the sum the passenger is charged after each trip.

Input

The first line of input contains integer number n ($1 \le n \le 10^5$) — the number of trips made by passenger.

Each of the following n lines contains the time of trip t_i ($0 \le t_i \le 10^9$), measured in minutes from the time of starting the system. All t_i are different, given in ascending order, i. e. $t_{i+1} > t_i$ holds for all $1 \le i < n$.

Output

Output n integers. For each trip, print the sum the passenger is charged after it.

Examples

input

```
3
10
20
30
```

output

```
20
20
10
```

input

```
10
13
45
46
60
103
115
126
150
256
516
```

output

```
20
20
10
0
20
0
0
20
20
10
```

Note

In the first example, the system works as follows: for the first and second trips it is cheaper to pay for two one-trip tickets, so each time 20 rubles is charged, after the third trip the system understands that it would be cheaper to buy a ticket for 90 minutes. This ticket costs 50 rubles, and the passenger had already paid 40 rubles, so it is necessary to charge 10 rubles only.

E. Nikita and stack

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nikita has a stack. A stack in this problem is a data structure that supports two operations. Operation `push(x)` puts an integer x on the top of the stack, and operation `pop()` deletes the top integer from the stack, i. e. the last added. If the stack is empty, then the operation `pop()` does nothing.

Nikita made m operations with the stack but forgot them. Now Nikita wants to remember them. He remembers them one by one, on the i-th step he remembers an operation he made p_i-th. In other words, he remembers the operations in order of some permutation $p_1, p_2, \ldots, p_m$. After each step Nikita wants to know what is the integer on the top of the stack after performing the operations he have already remembered, in the corresponding order. Help him!

Input

The first line contains the integer m ($1 \le m \le 10^5$) — the number of operations Nikita made.

The next m lines contain the operations Nikita remembers. The i-th line starts with two integers p_i and t_i ($1 \le p_i \le m$, $t_i = 0$ or $t_i = 1$) — the index of operation he remembers on the step i, and the type of the operation. t_i equals 0, if the operation is `pop()`, and 1, is the operation is `push(x)`. If the operation is `push(x)`, the line also contains the integer x_i ($1 \le x_i \le 10^6$) — the integer added to the stack.

It is guaranteed that each integer from 1 to m is present exactly once among integers p_i.

Output

Print m integers. The integer i should equal the number on the top of the stack after performing all the operations Nikita remembered on the steps from 1 to i. If the stack is empty after performing all these operations, print `-1`.

Examples

input

```
2
2 1 2
1 0
```

output

```
2
2
```

input

```
3
1 1 2
2 1 3
3 0
```

output

```
2
3
2
```

input

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

output

```
-1
-1
-1
-1
2
```

Note

In the first example, after Nikita remembers the operation on the first step, the operation `push(2)` is the only operation, so the answer is 2. After he remembers the operation `pop()` which was done before `push(2)`, answer stays the same.

In the second example, the operations are `push(2)`, `push(3)` and `pop()`. Nikita remembers them in the order they were performed.

In the third example Nikita remembers the operations in the reversed order.

F. Bacterial Melee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Julia is conducting an experiment in her lab. She placed several luminescent bacterial colonies in a horizontal testtube. Different types of bacteria can be distinguished by the color of light they emit. Julia marks types of bacteria with small Latin letters "a", ..., "z".

The testtube is divided into n consecutive regions. Each region is occupied by a single colony of a certain bacteria type at any given moment. Hence, the population of the testtube at any moment can be described by a string of n Latin characters.

Sometimes a colony can decide to conquer another colony in one of the adjacent regions. When that happens, the attacked colony is immediately eliminated and replaced by a colony of the same type as the attacking colony, while the attacking colony keeps its type. Note that a colony can only attack its neighbours within the boundaries of the testtube. At any moment, at most one attack can take place.

For example, consider a testtube with population "babb". There are six options for an attack that may happen next:

- the first colony attacks the second colony ($1 \to 2$), the resulting population is "bbbb";
- $2 \to 1$, the result is "aabb";
- $2 \to 3$, the result is "baab";
- $3 \to 2$, the result is "bbbb" (note that the result is the same as the first option);
- $3 \to 4$ or $4 \to 3$, the population does not change.

The pattern of attacks is rather unpredictable. Julia is now wondering how many different configurations of bacteria in the testtube she can obtain after a sequence of attacks takes place (it is possible that no attacks will happen at all). Since this number can be large, find it modulo $10^9 + 7$.

Input

The first line contains an integer n — the number of regions in the testtube ($1 \le n \le 5\,000$).

The second line contains n small Latin letters that describe the initial population of the testtube.

Output

Print one number — the answer to the problem modulo $10^9 + 7$.

Examples

input
```
3
aaa
```
output
```
1
```

input
```
2
ab
```
output
```
3
```

input
```
4
babb
```
output
```
11
```

input
```
7
abacaba
```
output
```
589
```

Note

In the first sample the population can never change since all bacteria are of the same type.

In the second sample three configurations are possible: "ab" (no attacks), "aa" (the first colony conquers the second colony), and "bb" (the second colony conquers the first colony).

To get the answer for the third sample, note that more than one attack can happen.