Educational Codeforces Round 93 (Rated for Div. 2)

A. Bad Triangle

1 second, 256 megabytes

You are given an array $a_1, a_2, \dots, a_n$, which is sorted in non-decreasing order ($a_i \le a_{i+1}$).

Find three indices i, j, k such that $1 \le i < j < k \le n$ and it is **impossible** to construct a non-degenerate triangle (a triangle with nonzero area) having sides equal to a_i, a_j and a_k (for example it is possible to construct a non-degenerate triangle with sides 3, 4 and 5 but impossible with sides 3, 4 and 7). If it is impossible to find such triple, report it.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains one integer n ($3 \le n \le 5 \cdot 10^4$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$; $a_{i-1} \le a_i$) — the array a.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case print the answer to it in one line.

If there is a triple of indices i, j, k ($i < j < k$) such that it is **impossible** to construct a non-degenerate triangle having sides equal to a_i, a_j and a_k, print that three indices in ascending order. If there are multiple answers, print any of them.

Otherwise, print `-1`.

input

```
3
7
4 6 11 11 15 18 20
4
10 10 10 11
3
1 1 1000000000
```

output

```
2 3 6
-1
1 2 3
```

In the first test case it is impossible with sides 6, 11 and 18. Note, that this is not the only correct answer.

In the second test case you always can construct a non-degenerate triangle.

B. Substring Removal Game

2 seconds, 256 megabytes

Alice and Bob play a game. They have a binary string s (a string such that each character in it is either 0 or 1). Alice moves first, then Bob, then Alice again, and so on.

During their move, the player can choose any number (not less than one) of **consecutive equal characters** in s and delete them.

For example, if the string is 10110, there are 6 possible moves (deleted characters are bold):

1. $\mathbf{1}0110 \to 0110$;
2. $1\mathbf{0}110 \to 1110$;
3. $10\mathbf{1}10 \to 1010$;
4. $101\mathbf{1}0 \to 1010$;
5. $10\mathbf{11}0 \to 100$;
6. $1011\mathbf{0} \to 1011$.

After the characters are removed, the characters to the left and to the right of the removed block become adjacent. I.e. the following sequence of moves is valid: $10\mathbf{11}0 \to 1\mathbf{00} \to 1$.

The game ends when the string becomes empty, and the score of each player is **the number of 1-characters deleted by them**.

Each player wants to maximize their score. Calculate the resulting score of Alice.

Input

The first line contains one integer T ($1 \le T \le 500$) — the number of test cases.

Each test case contains exactly one line containing a binary string s ($1 \le |s| \le 100$).

Output

For each test case, print one integer — the resulting score of Alice (the number of 1-characters deleted by her).

input

```
5
01111001
0000
111111
101010101
011011110111
```

output

```
4
0
6
3
6
```

Questions about the optimal strategy will be ignored.

C. Good Subarrays

2 seconds, 256 megabytes

You are given an array $a_1, a_2, \dots, a_n$ consisting of integers from 0 to 9. A subarray $a_l, a_{l+1}, a_{l+2}, \dots, a_{r-1}, a_r$ is good if the sum of elements of this subarray is equal to the length of this subarray ($\sum\limits_{i=l}^{r} a_i = r - l + 1$).

For example, if $a = [1, 2, 0]$, then there are 3 good subarrays: $a_{1 \dots 1} = [1]$, $a_{2 \dots 3} = [2, 0]$ and $a_{1 \dots 3} = [1, 2, 0]$.

Calculate the number of good subarrays of the array a.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains one integer n ($1 \le n \le 10^5$) — the length of the array a.

The second line of each test case contains a string consisting of n decimal digits, where the i-th digit is equal to the value of a_i.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case print one integer — the number of good subarrays of the array a.

input

```
3
3
120
5
11011
6
600005
```

output

```
3
6
1
```

The first test case is considered in the statement.

In the second test case, there are 6 good subarrays: $a_{1\dots1}$, $a_{2\dots2}$, $a_{1\dots2}$, $a_{4\dots4}$, $a_{5\dots5}$ and $a_{4\dots5}$.

In the third test case there is only one good subarray: $a_{2\dots6}$.

D. Colored Rectangles

2 seconds, 256 megabytes

You are given three multisets of pairs of colored sticks:

- R pairs of red sticks, the first pair has length r_1, the second pair has length r_2, $\dots$, the R-th pair has length r_R;
- G pairs of green sticks, the first pair has length g_1, the second pair has length g_2, $\dots$, the G-th pair has length g_G;
- B pairs of blue sticks, the first pair has length b_1, the second pair has length b_2, $\dots$, the B-th pair has length b_B;

You are constructing rectangles from these pairs of sticks with the following process:

1. take a pair of sticks of one color;
2. take a pair of sticks of another color different from the first one;
3. add the area of the resulting rectangle to the total area.

Thus, you get such rectangles that their opposite sides are the same color and their adjacent sides are not the same color.

Each pair of sticks can be used at most once, some pairs can be left unused. You are not allowed to split a pair into independent sticks.

What is the maximum area you can achieve?

Input

The first line contains three integers R, G, B ($1 \le R, G, B \le 200$) — the number of pairs of red sticks, the number of pairs of green sticks and the number of pairs of blue sticks.

The second line contains R integers $r_1, r_2, \dots, r_R$ ($1 \le r_i \le 2000$) — the lengths of sticks in each pair of red sticks.

The third line contains G integers $g_1, g_2, \dots, g_G$ ($1 \le g_i \le 2000$) — the lengths of sticks in each pair of green sticks.

The fourth line contains B integers $b_1, b_2, \dots, b_B$ ($1 \le b_i \le 2000$) — the lengths of sticks in each pair of blue sticks.

Output

Print the maximum possible total area of the constructed rectangles.

input

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

output

```
20
```

input

```
2 1 3
9 5
1
2 8 5
```

output

```
99
```

input

```
10 1 1
11 7 20 15 19 14 2 4 13 14
8
11
```

output

```
372
```

In the first example you can construct one of these rectangles: red and green with sides 3 and 5, red and blue with sides 3 and 4 and green and blue with sides 5 and 4. The best area of them is $4 \times 5 = 20$.

In the second example the best rectangles are: red/blue 9×8, red/blue 5×5, green/blue 2×1. So the total area is $72 + 25 + 2 = 99$.

In the third example the best rectangles are: red/green 19×8 and red/blue 20×11. The total area is $152 + 220 = 372$. Note that you can't construct more rectangles because you are not allowed to have both pairs taken to be the same color.

E. Two Types of Spells

3.5 seconds, 256 megabytes

Polycarp plays a computer game (yet again). In this game, he fights monsters using magic spells.

There are two types of spells: *fire* spell of power x deals x damage to the monster, and *lightning* spell of power y deals y damage to the monster and **doubles** the damage of the next spell Polycarp casts. Each spell can be cast **only once per battle**, but Polycarp can cast them in any order.

For example, suppose that Polycarp knows three spells: a fire spell of power 5, a lightning spell of power 1, and a lightning spell of power 8. There are 6 ways to choose the order in which he casts the spells:

- first, second, third. This order deals $5 + 1 + 2 \cdot 8 = 22$ damage;
- first, third, second. This order deals $5 + 8 + 2 \cdot 1 = 15$ damage;
- second, first, third. This order deals $1 + 2 \cdot 5 + 8 = 19$ damage;
- second, third, first. This order deals $1 + 2 \cdot 8 + 2 \cdot 5 = 27$ damage;
- third, first, second. This order deals $8 + 2 \cdot 5 + 1 = 19$ damage;
- third, second, first. This order deals $8 + 2 \cdot 1 + 2 \cdot 5 = 20$ damage.

Initially, Polycarp knows 0 spells. His spell set changes n times, each time he either learns a new spell or forgets an already known one. After each change, calculate the maximum possible damage Polycarp may deal using the spells he knows.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of changes to the spell set.

Each of the next n lines contains two integers tp and d ($0 \le tp_i \le 1$; $-10^9 \le d \le 10^9$; $d_i \neq 0$) — the description of the change. If tp_i if equal to 0, then Polycarp learns (or forgets) a fire spell, otherwise he learns (or forgets) a lightning spell.

If $d_i > 0$, then Polycarp learns a spell of power d_i. Otherwise, Polycarp forgets a spell with power $-d_i$, and it is guaranteed that he knew that spell before the change.

It is guaranteed that the powers of all spells Polycarp knows after each change are different (Polycarp never knows two spells with the same power).

Output

After each change, print the maximum damage Polycarp can deal with his current set of spells.

input
```
6
1 5
0 10
1 -5
0 5
1 11
0 -10
```
output
```
5
25
10
15
36
21
```

F. Controversial Rounds

2 seconds, 256 megabytes

Alice and Bob play a game. The game consists of several sets, and each set consists of several rounds. Each round is won either by Alice or by Bob, and the set ends when one of the players has won x rounds in a row. For example, if Bob won five rounds in a row and $x = 2$, then two sets ends.

You know that Alice and Bob have already played n rounds, and you know the results of some rounds. For each x from 1 to n, calculate the maximum possible number of sets that could have already finished if each set lasts until one of the players wins x rounds in a row. It is possible that the last set is still not finished — in that case, you should not count it in the answer.

Input

The first line contains one integer n ($1 \le n \le 10^6$) — the number of rounds.

The second line contains one string s of length n — the descriptions of rounds. If the i-th element of the string is `0`, then Alice won the i-th round; if it is `1`, then Bob won the i-th round, and if it is `?`, then you don't know who won the i-th round.

Output

In the only line print n integers. The i-th integer should be equal to the maximum possible number of sets that could have already finished if each set lasts until one of the players wins i rounds in a row.

input
```
6
11?000
```
output
```
6 3 2 1 0 0
```

input
```
5
01?01
```
output
```
5 1 0 0 0
```

input
```
12
???1??????1?
```
output
```
12 6 4 3 2 2 1 1 1 1 1 1
```

Let's consider the first test case:

- if $x = 1$ and $s = 110000$ or $s = 111000$ then there are six finished sets;
- if $x = 2$ and $s = 110000$ then there are three finished sets;
- if $x = 3$ and $s = 111000$ then there are two finished sets;
- if $x = 4$ and $s = 110000$ then there is one finished set;
- if $x = 5$ then there are no finished sets;
- if $x = 6$ then there are no finished sets.

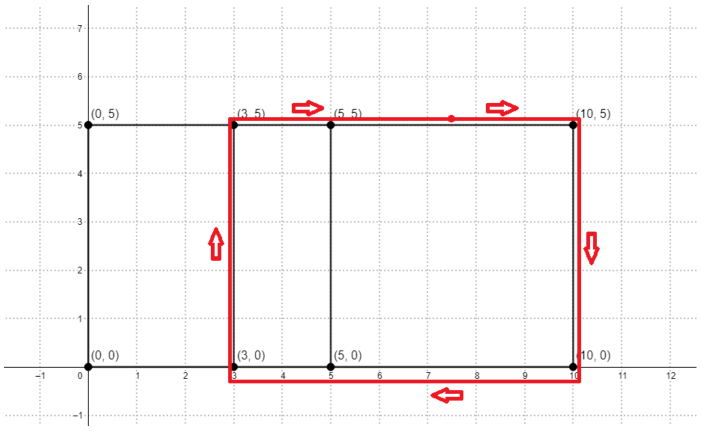
G. Running Competition

2 seconds, 512 megabytes

A running competition is going to be held soon. The stadium where the competition will be held can be represented by several segments on the coordinate plane:

- two horizontal segments: one connecting the points $(0, 0)$ and $(x, 0)$, the other connecting the points $(0, y)$ and (x, y);
- $n + 1$ vertical segments, numbered from 0 to n. The i-th segment connects the points $(a_i, 0)$ and (a_i, y); $0 = a_0 < a_1 < a_2 < \dots < a_{n-1} < a_n = x$.

For example, here is a picture of the stadium with $x = 10$, $y = 5$, $n = 3$ and $a = [0, 3, 5, 10]$:



A *lap* is a route that goes along the segments, starts and finishes at the same point, and never intersects itself (the only two points of a lap that coincide are its starting point and ending point). The length of a lap is a total distance travelled around it. For example, the red route in the picture representing the stadium is a lap of length 24.

The competition will be held in q stages. The i-th stage has length l_i, and the organizers want to choose a lap for each stage such that the length of the lap is a **divisor of** l_i. The organizers don't want to choose short laps for the stages, so for each stage, they want to find the maximum possible length of a suitable lap.

Help the organizers to calculate the maximum possible lengths of the laps for the stages! In other words, for every l_i, find the maximum possible integer L such that $l_i \bmod L = 0$, and there exists a lap of length **exactly** L.

If it is impossible to choose such a lap then print -1.

Input

The first line contains three integers n, x and y ($1 \le n, x, y \le 2 \cdot 10^5$, $n \le x$).

The second line contains $n + 1$ integers $a_0, a_1, \dots, a_n$ ($0 = a_0 < a_1 < a_2 < \dots < a_{n-1} < a_n = x$).

The third line contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of stages.

The fourth line contains q **even** integers $l_1, l_2, \dots, l_q$ ($4 \le l_i \le 10^6$) — the lengths of the stages.

Output

Print q numbers. The i-th number should be equal to the maximum possible length of a suitable lap for the i-th stage, or -1 if it is impossible to choose a lap for that stage.

input

```
3 10 5
0 3 5 10
6
24 30 14 16 18 10
```

output

```
24 30 14 16 -1 -1
```