Codeforces Round #205 (Div. 2)

A. Domino

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

Valera has got n domino pieces in a row. Each piece consists of two halves — the upper one and the lower one. Each of the halves contains a number from 1 to 6. Valera loves even integers very much, so he wants the sum of the numbers on the upper halves and the sum of the numbers on the lower halves to be even.

To do that, Valera can rotate the dominoes by 180 degrees. After the rotation the upper and the lower halves swap places. This action takes one second. Help Valera find out the minimum time he must spend rotating dominoes to make his wish come true.

Input

The first line contains integer n $(1 \le n \le 100)$, denoting the number of dominoes Valera has. Next n lines contain two space-separated integers x_i, y_i $(1 \le x_i, y_i \le 6)$. Number x_i is initially written on the upper half of the i-th domino, y_i is initially written on the lower half.

Output

Print a single number — the minimum required number of seconds. If Valera can't do the task in any time, print -1.

Examples

input

```
2
4 2
6 4
```

output

```
0
```

input

```
1
2 3
```

output

```
-1
```

input

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

output

```
1
```

Note

In the first test case the sum of the numbers on the upper halves equals 10 and the sum of the numbers on the lower halves equals 6. Both numbers are even, so Valera doesn't required to do anything.

In the second sample Valera has only one piece of domino. It is written 3 on the one of its halves, therefore one of the sums will always be odd.

In the third case Valera can rotate the first piece, and after that the sum on the upper halves will be equal to 10, and the sum on the lower halves will be equal to 8.

B. Two Heaps

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

Valera has $2 \cdot n$ cubes, each cube contains an integer from 10 to 99. He arbitrarily chooses n cubes and puts them in the first heap. The remaining cubes form the second heap.

Valera decided to play with cubes. During the game he takes a cube from the first heap and writes down the number it has. Then he takes a cube from the second heap and write out its two digits near two digits he had written (to the right of them). In the end he obtained a single fourdigit integer — the first two digits of it is written on the cube from the first heap, and the second two digits of it is written on the second cube from the second heap.

Valera knows arithmetic very well. So, he can easily count the number of distinct fourdigit numbers he can get in the game. The other question is: how to split cubes into two heaps so that this number (the number of distinct fourdigit integers Valera can get) will be as large as possible?

Input

The first line contains integer n $(1 \le n \le 100)$. The second line contains $2 \cdot n$ space-separated integers a_i $(10 \le a_i \le 99)$, denoting the numbers on the cubes.

Output

In the first line print a single number — the maximum possible number of distinct four-digit numbers Valera can obtain. In the second line print $2 \cdot n$ numbers b_i $(1 \le b_i \le 2)$. The numbers mean: the i-th cube belongs to the b_i-th heap in your division.

If there are multiple optimal ways to split the cubes into the heaps, print any of them.

Examples

input

```
1
10 99
```

output

```
1
2 1
```

input

```
2
13 24 13 45
```

output

```
4
1 2 2 1
```

Note

In the first test case Valera can put the first cube in the first heap, and second cube — in second heap. In this case he obtain number 1099. If he put the second cube in the first heap, and the first cube in the second heap, then he can obtain number 9910. In both cases the maximum number of distinct integers is equal to one.

In the second test case Valera can obtain numbers $1313, 1345, 2413, 2445$. Note, that if he put the first and the third cubes in the first heap, he can obtain only two numbers 1324 and 1345.

C. Find Maximum

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

Valera has array a, consisting of n integers $a_0, a_1, ..., a_{n-1}$, and function $f(x)$, taking an integer from 0 to $2^n - 1$ as its single argument. Value $f(x)$ is calculated by formula $f(x) = \sum_{i=0}^{n-1} a_i \cdot bit(i)$, where value $bit(i)$ equals one if the binary representation of number x contains a 1 on the i-th position, and zero otherwise.

For example, if $n = 4$ and $x = 11$ ($11 = 2^0 + 2^1 + 2^3$), then $f(x) = a_0 + a_1 + a_3$.

Help Valera find the maximum of function $f(x)$ among all x, for which an inequality holds: $0 \le x \le m$.

Input

The first line contains integer n ($1 \le n \le 10^5$) — the number of array elements. The next line contains n space-separated integers $a_0, a_1, ..., a_{n-1}$ ($0 \le a_i \le 10^4$) — elements of array a.

The third line contains a sequence of digits zero and one without spaces $s_0s_1... s_{n-1}$ — the binary representation of number m. Number m equals $\sum_{i=0}^{n-1} 2^i \cdot s_i$.

Output

Print a single integer — the maximum value of function $f(x)$ for all $x \in [0..m]$.

Examples

input

```
2
3 8
10
```

output

```
3
```

input

```
5
17 0 10 2 1
11010
```

output

```
27
```

Note

In the first test case $m = 2^0 = 1$, $f(0) = 0$, $f(1) = a_0 = 3$.

In the second sample $m = 2^0 + 2^1 + 2^3 = 11$, the maximum value of function equals $f(5) = a_0 + a_2 = 17 + 10 = 27$.

D. Queue

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

There are n schoolchildren, boys and girls, lined up in the school canteen in front of the bun stall. The buns aren't ready yet and the line is undergoing some changes.

Each second all boys that stand right in front of girls, simultaneously swap places with the girls (so that the girls could go closer to the beginning of the line). In other words, if at some time the i-th position has a boy and the $(i+1)$-th position has a girl, then in a second, the i-th position will have a girl and the $(i+1)$-th one will have a boy.

Let's take an example of a line of four people: a boy, a boy, a girl, a girl (from the beginning to the end of the line). Next second the line will look like that: a boy, a girl, a boy, a girl. Next second it will be a girl, a boy, a girl, a boy. Next second it will be a girl, a girl, a boy, a boy. The line won't change any more.

Your task is: given the arrangement of the children in the line to determine the time needed to move all girls in front of boys (in the example above it takes 3 seconds). Baking buns takes a lot of time, so no one leaves the line until the line stops changing.

Input

The first line contains a sequence of letters without spaces $s_1 s_2 \ldots s_n$ $(1 \le n \le 10^6)$, consisting of capital English letters `M` and `F`. If letter s_i equals `M`, that means that initially, the line had a boy on the i-th position. If letter s_i equals `F`, then initially the line had a girl on the i-th position.

Output

Print a single integer — the number of seconds needed to move all the girls in the line in front of the boys. If the line has only boys or only girls, print 0.

Examples

input
```
MFM
```
output
```
1
```

input
```
MMFF
```
output
```
3
```

input
```
FFMMM
```
output
```
0
```

Note

In the first test case the sequence of changes looks as follows: `MFM` $\to$ `FMM`.

The second test sample corresponds to the sample from the statement. The sequence of changes is: `MMFF` $\to$ `MFMF` $\to$ `FMFM` $\to$ `FFMM`.

E. Antichain

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

You have a directed acyclic graph G, consisting of n vertexes, numbered from 0 to $n - 1$. The graph contains n edges numbered from 0 to $n - 1$. An edge with number i connects vertexes i and $(i + 1) \bmod n$, and it can be directed in either direction (from i to $(i + 1) \bmod n$, or vise versa).

Operation $x \bmod y$ means taking the remainder after dividing number x by number y.

Let's call two vertexes u and v in graph G comparable if the graph contains a path either from u to v or from v to u. We'll assume that an antichain is a set of vertexes of graph G, where any two distinct vertexes are not comparable. The size of an antichain is the number of vertexes in the corresponding set. An antichain is maximum if the graph doesn't have antichains of a larger size.

Your task is to find the size of the maximum antichain in graph G.

Input

The first line contains the sequence of characters $s_0s_1... s_{n-1}$ $(2 \le n \le 10^6)$, consisting of numbers zero and one. The length of the line (number n) corresponds to the number of vertexes and edges in graph G. If character s_i $(i \ge 0)$ equals 0, then the edge between vertexes i and $(i + 1) \bmod n$ is directed from the i-th vertex to the $(i + 1) \bmod n$-th one, otherwise — to the opposite point.

It is guaranteed that the given graph is acyclic.

Output

Print a single integer — the size of the maximum antichain of graph G.

Examples

input

```
001
```

output

```
1
```

input

```
110010
```

output

```
3
```

Note

Consider the first test sample. The graph's G edges are: $0 \to 1$, $1 \to 2$, $0 \to 2$. We can choose the set of vertexes $[0]$ as the maximum antichain. We cannot choose an antichain of larger size.