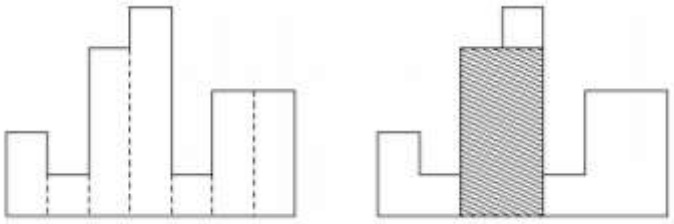
Training 2 - DST - 20211

A. 02. HIST

1 second, 256 megabytes

Lỗ Ban là một vị thợ mộc nổi tiếng bậc nhất thời Tống của Trung Quốc với đôi bàn tay cực kỳ tài hoa khéo léo. Ngưỡng mộ tài năng của Lỗ Ban, Vua Tống mời Lỗ Ban vào triều làm quan chuyên quản lý việc thiết kế cung điện và chế tác vật dụng tinh xảo. Vua Tống trong một chuyến du ngoạn ở núi Ngũ Nhạc tình cờ phát hiện một phiến đá ngũ sắc tuyệt đẹp, Vua Tống nảy ra ý định đem phiến đá này về cung để chế tác thành một bàn cờ. Nhưng Vua Tống nhanh chóng phát hiện ra rằng, phiến đá này có hình dạng kỳ lạ rất khó để có thể cắt ra phần diện tích vuông vắn đủ lớn cho bàn cờ. Vậy là Vua Tống triệu Lỗ Ban vào triều để thương lượng:

Bề mặt phiến đá có thể được mô tả như là một hình đa giác được ghép thành từ nhiều phiến đá nhỏ hình chữ nhật có chung nhau một mép, có các chiều dài khác nay nhưng giống nhau về chiều rộng và bằng 1 đơn vị. Trong hình vẽ dưới đây, phiến đá đa giác gồm các hình chữ nhật có chiều cao lần lượt từ trái qua phải là 2,1,4,5,1,3,3 và chiều rộng đều bằng 1.



Yêu cầu: Bạn cần giúp Lỗ Ban tìm ra hình chữ nhật chung mép với các hình chữ nhật nhỏ và có diện tích lớn nhất nằm trong phiến đá đa giác nói trên. Ở hình vẽ dưới, hình chữ nhật lớn nhất là hình được gạch chéo.

Input

Chứa một hoặc nhiều test. Mỗi test mô tả một đa giác bắt đầu bằng số nguyên n $(1 \leq n \leq 1000000)$ là số lượng hình chữ nhật nhỏ cấu thành đa giác. Tiếp theo sau là n số nguyên $l_1, l_2, \ldots, l_n$ với $0 \leq l_i \leq 100000000$ lần lượt từ trái sang phải biểu thị chiều dài của các hình chữ nhật. Chiều rộng của các hình chữ nhật bằng nhau và bằng 1. File kết thúc với dòng ghi duy nhất một số 0.

Output

Với mỗi test ghi trên một dòng diện tích của hình chữ nhật nằm trong đa giác thỏa mãn điều kiện đề bài.

input
2 0 0 3 0 0 0 4 0 1 0 1 5 1 0 1 0 1 6 2 0 1 0 1 0 0

output
0 0 1 1 2

B. 02. POSTMAN

1 second, 256 megabytes

Chuyển phát hàng là một công việc quan trọng trong thương mại điện tử là lĩnh vực phát triển bùng nổ trong thời gian hiện nay. Ta xét công việc của một nhân viên giao hàng của Công ty XYZ chuyên bán hàng trên mạng. Nhân viên giao hàng cần phát các kiện hàng (được đóng gói trong các hộp cùng kích thước) đến các khách hàng có địa chỉ trên một đại lộ có dạng một đường thẳng.

Nhân viên giao hàng sẽ nhận các kiện hàng tại trụ sở công ty ở vị trí $x = 0$, và cần chuyển phát hàng đến n khách hàng, được đánh số từ 1 đến n. Biết x_i và m_i là vị trí của khách hàng i và số lượng kiện hàng cần chuyển cho khách hàng này. Do các kiện hàng là khá cồng kềnh nên mỗi lần đi chuyển phát nhân viên giao hàng chỉ có thể mang theo không quá k kiện hàng.

Nhân viên giao hàng xuất phát từ trụ sở, nhận một số (không quá k) kiện hàng và di chuyển theo đại lộ để chuyển phát cho một số khách hàng. Khi giao hết các kiện hàng mang theo, nhân viên giao hàng lại quay trở về trụ sở và lặp lại công việc nói trên cho đến khi chuyển phát được tất cả các kiện hàng cho khách hàng. Sau khi kết thúc công việc chuyển phát, nhân viên phải quay trở lại trụ sở công ty để nộp cho phòng kế toán tất cả các hóa đơn giao nhận có ký nhận của khách hàng. Giả thiết là: tốc độ di chuyển của nhân viên là 1 đơn vị khoảng cách trên một đơn vị thời gian. Thời gian nhận hàng ở trụ sở công ty và thời gian bàn giao hàng cho khách hàng được coi là bằng 0.

Yêu cầu: Giả sử thời điểm mà nhân viên giao hàng bắt đầu công việc là 0. Hãy giúp nhân viên giao hàng tìm cách hoàn thành công việc đã mô tả ở trên tại thời điểm sớm nhất.

Input
Dòng đầu tiên chứa hai số nguyên dương được ghi cách nhau bởi dấu cách n và k $(n \leq 1000; k \leq 10^7)$.

Dòng thứ i trong số n dòng tiếp theo chứa hai số nguyên được ghi cách nhau bởi dấu cách x_i $(|xi| \leq 10^7)$ và $m_i (1 \leq m_i \leq 10^7)$

Output
Ghi ra một số nguyên là thời điểm sớm nhất mà người giao hàng có thể hoàn thành nhiệm vụ của mình.

input

```
7 1
9400000 10000000
9500000 10000000
9600000 10000000
9700000 10000000
9800000 10000000
9900000 10000000
10000000 10000000
```

output

```
1358000000000000
```

input

```
4 10
-7 5
-2 3
5 7
9 5
```

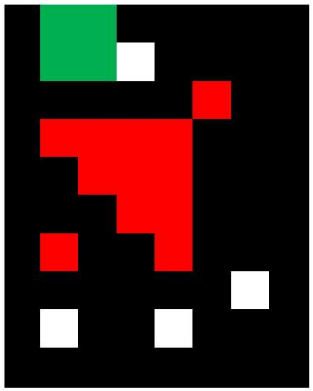
output

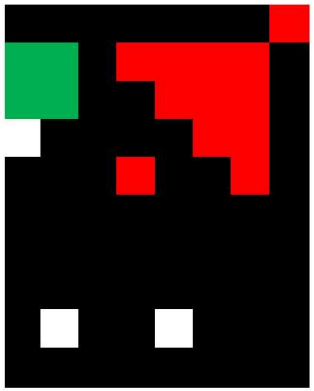
```
42
```

C. 02. LOCATE

1 second, 256 megabytes

Hùng là tướng quân của một đội quân lớn, bộ phận thông tin đã chỉ ra rằng kẻ địch đã khởi động một loạt những máy bay chiến đấu nhỏ tiến sát tấn công quân của Hùng. Chỉ còn rất ít thời gian để phát hiện ra vị trí chính xác của các chiến đấu cơ này để mà tiêu diệt chúng. Hùng đã điều chỉnh lên tối đa độ nhạy của rada để định vị các chiến đấu cơ này dù rằng kích thước của chúng rất nhỏ, tuy nhiên điều đó cũng định vị luôn cả những chú chim đang bay gần đó. Tuy vậy Hùng biết rằng tất cả các chiến đấu cơ dịch chuyển chính xác theo cùng một cách, điều đó hi vọng giúp Hùng định vị chính xác vị trí của đàn chiến đấu cơ. Cho biết mô tả của 2 hình ảnh mà rada thu được cách nhau 1 phút, Hùng muốn định vị tập lớn nhất các điểm trên ảnh 1, mà có thể tìm thấy nó trên ảnh 2 sau một khoảng di chuyển xác định. Biết rằng dàn chiến đấu cơ có thể không xuất hiện ở trên cả hai ảnh rada.





Input

Dòng đầu tiên ghi ra một số nguyên T là số test. Với mỗi test:

Dòng đầu tiên chứa 2 số nguyên L và C tương ứng là số dòng và số cột của 2 ảnh ($1 \leq L, C \leq 1000$). $2 * L$ dòng tiếp theo mỗi dòng chứa C số nguyên 0 hoặc 1 cách nhau bởi dấu cách, trong đó L dòng đầu mô tả ảnh 1, L dòng sau mô tả ảnh 2. Ở 1 vị trí xác định, 1 tượng trưng cho có chiến đấu cơ hoặc chim ở đó còn 0 tượng trưng cho vị trí đó không có vật thể nào. Biết rằng số lượng số 1 trong một ảnh rada không quá 10000.

Output

Với mỗi test tương ứng, ghi duy nhất một số nguyên N là số điểm lớn nhất các đối tượng dịch chuyển theo cùng một cách thức.

input

```
1
10 8
0 1 1 0 0 0 0 0
0 1 1 1 0 0 0 0
0 0 0 0 0 1 0 0
0 1 1 1 1 0 0 0
0 0 1 1 1 0 0 0
0 0 0 1 1 0 0 0
0 1 0 0 1 0 0 0
0 0 0 0 0 0 1 0
0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
1 1 0 1 1 1 1 0
1 1 0 0 1 1 1 0
1 0 0 0 0 1 1 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0
```

output

```
12
```

D. 02. WATERJUG BFS

1 second, 256 megabytes

There are two jugs, a-litres jug and b-litres jug (a, b are positive integers). There is a pump with unlimited water. Given a positive integer c, how to get exactly c litres.

Example: with $a = 6$, $b = 8$, and $c = 4$, we perform 4 operations below to get exactly 4 litres (a state is represented by (x, y) in which x and y are respectively the amount of water in the 6-litres jug and the 8-litres jug):

- Fill the 6-litres jug, we get (6,0)
- Pour the 6-litres jug to the 8-litres jug, we get (0,6)
- Fill the 6-litres jug, we get (6,6)

- Pour the 6-litres jug to the 8-litres jug, we get (4,8)

Input
Unique line contains positive integers a, b, c ($1 \leq a, b, c \leq 900$).

Output
Line contains the minimal number of steps to get c litres or -1 if no solution found.

input
6 8 4
output
4

E. 02.REROAD

1 second, 256 megabytes

Đường vành đai III của thành phố Naho nổi tiếng về chất lượng mặt đường tồi tệ. Lí do là đội ngũ sửa chữa đường của thành phố này quá tùy tiện. Đường vành đai III được chia nhỏ thành N đoạn kế tiếp nhau có cùng chiều dài đơn vị. Mỗi lần sửa chữa đường họ tiến hành như sau: Một nhóm công nhân sẽ lựa chọn một đoạn đường nào đó và thay thế toàn bộ lớp nhựa phủ đường trên đoạn đó. Loại nhựa đường được thay thế trên đoạn này có thể khác hẳn với loại nhựa đường trên các đoạn khác làm gây khó khăn cho việc đi lại trên đường. Là cư dân thành phố Naho và là một lập trình viên giỏi, Hải quyết định sử dụng hiểu biết của mình để giúp ích cho xã hội và làm thuận tiện cho cuộc sống của người dân thành phố khi phải đi qua đường vành đai III. Cụ thể là Hải quyết định tạo trang web chứa thông tin về độ gập ghềnh của đường. Hải đánh số các đoạn đường từ 1 đến N và thu thập thông tin về loại nhựa đường trên từng đoạn $t_1, t_2, \ldots, t_N$ (t_i là mã loại nhựa đường phủ trên đoạn đường thứ i). Hải định nghĩa một phần đường là một dãy liên tục các đoạn đường được phủ cùng loại nhựa phủ t_k và bên trái và bên phải phần đường đó là các đoạn đường (nếu tồn tại) được phủ loại nhựa khác. Cuối cùng, Hải xác định độ gập ghềnh của đường bằng tổng số lượng phần đường trên đường vành đai III. Ví dụ đường phố chứa các đoạn đường được phủ bởi loại nhựa có mã lần lượt tương ứng với 1,1,0,1,1,1 sẽ có độ gập ghềnh bằng 3 vì nó chứa đúng 3 phần đường 11, 0 và 111. Đường phố chứa các đoạn đường được phủ bởi loại nhựa có mã lần lượt tương ứng với 2,2,2,2 là lý tưởng vì nó chỉ chứa 1 phần đường và có độ gập ghềnh đúng bằng 1.

Dân chúng sẽ hài lòng nếu Hải luôn có thể tính toán và cung cấp trên trang web độ gập ghềnh của đường tại thời điểm hiện tại. Đáng tiếc là mặt đường được thay đổi khá thường xuyên và Hải không muốn mỗi lần như vậy lại phải ra đường thu thập dữ liệu. Vì vậy Hải yêu cầu đội ngũ sửa chữa đường mỗi lần sửa đường phải gửi một thông báo cho Hải. Mỗi thông báo bao gồm 2 số là số thứ tự đoạn đường được sửa và mã loại nhựa được phủ mới. Nhiệm vụ của Hải là phải cập nhật độ gập ghềnh thực tế của đường sau mỗi thông báo như vậy.

Input
Dòng đầu tiên chứa số tự nhiên duy nhất N là số lượng đoạn đường ($1 \leq \mathrm{N} \leq 10^5$).

Dòng tiếp theo chứa N số nguyên $t_1, t_2, \ldots, t_N$ là các loại nhựa đường ban đầu phủ trên các đoạn đường ($|t_i| \leq 10^9$).

Dòng thứ 3 chứa số nguyên duy nhất Q là số lượng thông báo từ dân chúng về việc sửa chữa mặt đường ($1 \leq Q \leq 10^5$).

Mỗi dòng trong số Q dòng tiếp theo chứa lần lượt các thông báo. Thông báo thứ i là cặp hai số nguyên p_i, c_i là số thứ tự của đoạn đường được sửa và mã loại nhựa đường mới được phủ lên trên đoạn đường này ($1 \leq p_i \leq N$, $|c_i| \leq 10^9$). Đoạn đường được đánh số từ 1 đến N theo đúng thứ tự ghi loại nhựa đường trong dòng thứ 2 của dữ liệu vào.

Output

In ra Q dòng: dòng thứ i ($1 \leq i \leq Q$) phải chứa đúng một số nguyên duy nhất là giá trị độ gập ghềnh của đường sau i thông báo sửa đường đầu tiên.

input

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

output

```
1
3
5
5
3
```

input

```
7
1 1 2 3 2 2 1
3
2 2
4 2
6 9
```

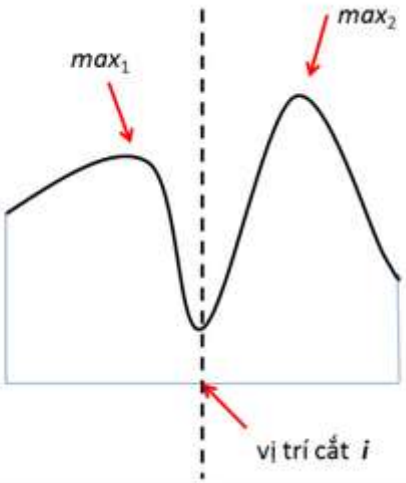
output

```
5
3
4
```

F. 02.SIGNAL

1 second, 256 megabytes

Một thiết bị cảm biến có nhiệm vụ thu nhận dữ liệu về các đối tượng trong 1 khu vực để truyền về cho trung tâm xử lý. Mỗi đối tượng sẽ được biểu diễn bởi 1 dãy số nguyên dương. Như vậy, cảm biến sẽ truyền các dãy số về cho trung tâm xử lý. Tuy nhiên, do các đối tượng ở gần nhau và có tín hiệu nhiễu nên một dãy số gửi về cho trung tâm có thể là dữ liệu của 2 đối tượng. Dãy $a_1, \ldots, a_n$ sẽ là dữ liệu của 2 đối tượng nếu có 1 vị trí i ($1 < i < n$) sao cho $max\{a_1, .., a_{i-1}\} - a_i \geq b$ và $max\{a_{i+1}, .., a_n\} - a_i \geq b$ với b là hằng số cho trước (xem minh hoạ trong Hình 1). Khi phát hiện một dãy số $a_1, \ldots, a_n$ là dữ liệu của 2 đối tượng thì cần phải tiến hành cắt dãy số đó thành 2 dãy, mỗi dãy là dữ liệu của một đối tượng. Khi đó vị trí cắt sẽ là vị trí i sao cho $max\{a_1, .., a_{i-1}\} - a_i + max\{a_{i+1}, .., a_n\} - a_i$ đạt giá trị lớn nhất (giá trị đó gọi là độ đo cắt tín hiệu).



Ví dụ: với giá trị b = 5 thì dãy số 3, 5, 4, 7, 2, 5, 4, 6, 9, 8 là dữ liệu của 2 đối tượng vì tìm thấy vị trí $i = 5$ tại đó $max\{3, 5, 4, 7\} - 2 \geq 5$ và $max\{5, 4, 6, 9, 8\} - 2 \geq 5$ và vị trí $i = 5$ cũng chính là vị trí cắt.

Yêu cầu: cho trước giá trị n, b và dãy số nguyên dương $a_1, \ldots, a_n$. Hãy lập trình kiểm tra xem dãy số $a_1, \ldots, a_n$ có phải là dữ liệu biểu diễn 2 đối tượng hay không và tính độ đo cắt tín hiệu nếu câu trả lời là có.

Input
Dữ liệu đầu vào bao gồm các dòng sau:

- Dòng thứ nhất chứa 2 số nguyên dương n và b ($3 \leq n \leq 200000, 1 \leq b \leq 50$)
- Dòng thứ 2 chứa n tự nhiên $a_1, \ldots, a_n$

Output
Ghi ra độ đo cắt tín hiệu nếu dãy đầu vào là dữ liệu biểu diễn 2 đối tượng và giá trị -1 nếu ngược lại.

input
10 5 3 5 4 7 2 5 4 6 9 8
output
12

G. SORT INT

2 seconds, 256 megabytes

Cho mảng các số nguyên a. Hãy sắp xếp mảng A theo thứ tự tăng dần

Input
Dòng đầu chứa một số nguyên là số phần tử của mảng $n \leq 10^6$.

Dòng thứ hai chứa n số nguyên.

Output
In ra trên một dòng mảng a theo thứ tự tăng dần.

input
5 7 4 1 2 3
output
1 2 3 4 7

H. PARENTHESES

1 second, 256 megabytes

Input
The input file consists of several datasets. The first line of the input file contains the number of datasets which is a positive integer T and is not greater than 1000. Each of T following lines describes a parentheses expression including: '(',')','[',']','{','}'.

Output
For each dataset, write in one line 1 or 0 if the expression is correct or not respectively.

input
2 ([]()) ()()()[}
output
1 0

I. Bracket Sequence

2 seconds, 256 megabytes

A *bracket sequence* is a string, containing only characters "(", ")", "[" and "]".

A *correct bracket sequence* is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example, bracket sequences "()[]", "([])" are correct (the resulting expressions are: "(1)+[1]", "([1+1]+1)"), and "][" and "[" are not. **The empty string is a correct bracket sequence by definition.**

A *substring* $s[l... r]$ $(1 \le l \le r \le |s|)$ of string $s = s_1 s_2 ... s_{|s|}$ (where $|s|$ is the length of string s) is the string $s_l s_{l+1} ... s_r$. **The empty string is a substring of any string by definition**.

You are given a bracket sequence, not necessarily correct. Find its substring which is a correct bracket sequence and contains as many opening square brackets «[» as possible.

Input

The first and the only line contains the bracket sequence as a string, consisting only of characters "(", ")", "[" and "]". It is guaranteed that the string is non-empty and its length doesn't exceed 10^5 characters.

Output

In the first line print a single integer — the number of brackets «[» in the required bracket sequence. In the second line print the optimal sequence. If there are more than one optimal solutions print any of them.

input
([])
output
1 ([])

input
(((
output
0

J. Cd and pwd commands

3 seconds, 256 megabytes

Vasya is writing an operating system shell, and it should have commands for working with directories. To begin with, he decided to go with just two commands: `cd` (change the current directory) and `pwd` (display the current directory).

Directories in Vasya's operating system form a traditional hierarchical tree structure. There is a single root directory, denoted by the slash character "/". Every other directory has a name — a non-empty string consisting of lowercase Latin letters. Each directory (except for the root) has a parent directory — the one that contains the given directory. It is denoted as "..".

The command `cd` takes a single parameter, which is a path in the file system. The command changes the current directory to the directory specified by the path. The path consists of the names of directories separated by slashes. The name of the directory can be "..", which means a step up to the parent directory. «..» can be used in any place of the path, maybe several times. If the path begins with a slash, it is considered to be an absolute path, that is, the directory changes to the specified one, starting from the root. If the parameter begins with a directory name (or ".."), it is considered to be a relative path, that is, the directory changes to the specified directory, starting from the current one.

The command `pwd` should display the absolute path to the current directory. This path must not contain "..".

Initially, the current directory is the root. All directories mentioned explicitly or passed indirectly within any command `cd` are considered to exist. It is guaranteed that there is no attempt of transition to the parent directory of the root directory.

Input

The first line of the input data contains the single integer n $(1 \le n \le 50)$ — the number of commands.

Then follow n lines, each contains one command. Each of these lines contains either command `pwd`, or command `cd`, followed by a space-separated non-empty parameter.

The command parameter `cd` only contains lower case Latin letters, slashes and dots, two slashes cannot go consecutively, dots occur only as the name of a parent pseudo-directory. The command parameter `cd` does not end with a slash, except when it is the only symbol that points to the root directory. The command parameter has a length from 1 to 200 characters, inclusive.

Directories in the file system can have the same names.

Output

For each command `pwd` you should print the full absolute path of the given directory, ending with a slash. It should start with a slash and contain the list of slash-separated directories in the order of being nested from the root to the current folder. It should contain no dots.

input

```
7
pwd
cd /home/vasya
pwd
cd ..
pwd
cd vasya/../petya
pwd
```

output

```
/
/home/vasya/
/home/
/home/petya/
```

input

```
4
cd /a/b
pwd
cd ../a/b
pwd
```

output

```
/a/b/
/a/a/b/
```

K. Queue

2 seconds, 256 megabytes

There are n walruses standing in a queue in an airport. They are numbered starting from the queue's tail: the 1-st walrus stands at the end of the queue and the n-th walrus stands at the beginning of the queue. The i-th walrus has the age equal to a_i.

The i-th walrus becomes displeased if there's a younger walrus standing in front of him, that is, if exists such j ($i < j$), that $a_i > a_j$. The displeasure of the i-th walrus is equal to the number of walruses between him and the furthest walrus ahead of him, which is younger than the i-th one. That is, the further that young walrus stands from him, the stronger the displeasure is.

The airport manager asked you to count for each of n walruses in the queue his displeasure.

Input

The first line contains an integer n ($2 \le n \le 10^5$) — the number of walruses in the queue. The second line contains integers a_i ($1 \le a_i \le 10^9$).

Note that some walruses can have the same age but for the displeasure to emerge the walrus that is closer to the head of the queue needs to be **strictly younger** than the other one.

Output

Print n numbers: if the i-th walrus is pleased with everything, print "-1" (without the quotes). Otherwise, print the i-th walrus's displeasure: the number of other walruses that stand between him and the furthest from him younger walrus.

input
6 10 8 5 3 50 45
output
2 1 0 -1 0 -1

input
7 10 4 6 3 2 8 15
output
4 2 1 0 -1 -1 -1

input
5 10 3 1 10 11
output
1 0 -1 -1 -1

L. Array

2 seconds, 256 megabytes

Vitaly has an array of n distinct integers. Vitaly wants to divide this array into three **non-empty** sets so as the following conditions hold:

1. The product of all numbers in the first set is less than zero (< 0).
2. The product of all numbers in the second set is greater than zero (> 0).
3. The product of all numbers in the third set is equal to zero.
4. Each number from the initial array must occur in exactly one set.

Help Vitaly. Divide the given array.

Input

The first line of the input contains integer n $(3 \le n \le 100)$. The second line contains n space-separated distinct integers $a_1, a_2, ..., a_n$ $(|a_i| \le 10^3)$ — the array elements.

Output

In the first line print integer n_1 $(n_1 > 0)$ — the number of elements in the first set. Then print n_1 numbers — the elements that got to the first set.

In the next line print integer n_2 $(n_2 > 0)$ — the number of elements in the second set. Then print n_2 numbers — the elements that got to the second set.

In the next line print integer n_3 $(n_3 > 0)$ — the number of elements in the third set. Then print n_3 numbers — the elements that got to the third set.

The printed sets must meet the described conditions. It is guaranteed that the solution exists. If there are several solutions, you are allowed to print any of them.

input
3 -1 2 0
output
1 -1 1 2 1 0

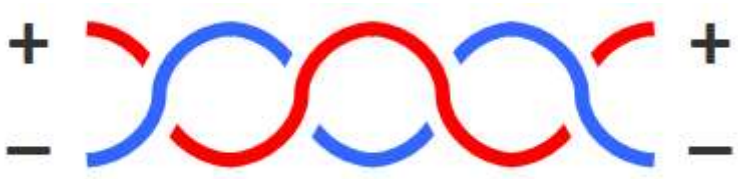
input
4 -1 -2 -3 0
output
1 -1 2 -3 -2 1 0

M. Alternating Current

1 second, 256 megabytes

Mad scientist Mike has just finished constructing a new device to search for extraterrestrial intelligence! He was in such a hurry to launch it for the first time that he plugged in the power wires without giving it a proper glance and started experimenting right away. After a while Mike observed that the wires ended up entangled and now have to be untangled again.

The device is powered by two wires "plus" and "minus". The wires run along the floor from the wall (on the left) to the device (on the right). Both the wall and the device have two contacts in them on the same level, into which the wires are plugged in some order. The wires are considered entangled if there are one or more places where one wire runs above the other one. For example, the picture below has four such places (top view):



Mike knows the sequence in which the wires run above each other. Mike also noticed that on the left side, the "plus" wire is always plugged into the top contact (as seen on the picture). He would like to untangle the wires without unplugging them and **without moving** the device. Determine if it is possible to do that. A wire can be freely moved and stretched on the floor, but cannot be cut.

To understand the problem better please read the notes to the test samples.

Input

The single line of the input contains a sequence of characters "+" and "−" of length n ($1 \le n \le 100000$). The i-th ($1 \le i \le n$) position of the sequence contains the character "+", if on the i-th step from the wall the "plus" wire runs above the "minus" wire, and the character "−" otherwise.

Output

Print either "`Yes`" (without the quotes) if the wires can be untangled or "`No`" (without the quotes) if the wires cannot be untangled.

input
-++-
output
Yes

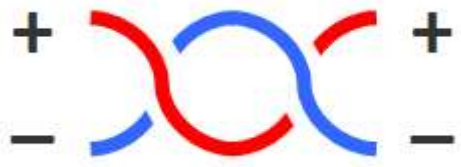
input
+-
output
No

input
++
output
Yes

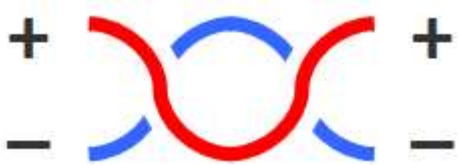
input
-
output
No

The first testcase corresponds to the picture in the statement. To untangle the wires, one can first move the "plus" wire lower, thus eliminating the two crosses in the middle, and then draw it under the "minus" wire, eliminating also the remaining two crosses.

In the second testcase the "plus" wire makes one full revolution around the "minus" wire. Thus the wires cannot be untangled:



In the third testcase the "plus" wire simply runs above the "minus" wire twice in sequence. The wires can be untangled by lifting "plus" and moving it higher:



In the fourth testcase the "minus" wire runs above the "plus" wire once. The wires cannot be untangled without moving the device itself:

O. Union of Doubly Linked Lists

2 seconds, 256 megabytes

Doubly linked list is one of the fundamental data structures. A doubly linked list is a sequence of elements, each containing information about the previous and the next elements of the list. In this problem all lists have linear structure. I.e. each element except the first has exactly one previous element, each element except the last has exactly one next element. The list is not closed in a cycle.

In this problem you are given n memory cells forming one or more doubly linked lists. Each cell contains information about element from some list. Memory cells are numbered from 1 to n.

For each cell i you are given two values:

- l_i — cell containing previous element for the element in the cell i;
- r_i — cell containing next element for the element in the cell i.

If cell i contains information about the element which has no previous element then $l_i = 0$. Similarly, if cell i contains information about the element which has no next element then $r_i = 0$.

Three lists are shown on the picture.

For example, for the picture above the values of l and r are the following: $l_1 = 4, r_1 = 7; l_2 = 5, r_2 = 0; l_3 = 0, r_3 = 0; l_4 = 6, r_4 = 1; l_5 = 0, r_5 = 2; l_6 = 0, r_6 = 4; l_7 = 1, r_7 = 0$.

Your task is to unite all given lists in a single list, joining them to each other in any order. In particular, if the input data already contains a single list, then there is no need to perform any actions. Print the resulting list in the form of values l_i, r_i.

Any other action, other than joining the beginning of one list to the end of another, can not be performed.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of memory cells where the doubly linked lists are located.

Each of the following n lines contains two integers l_i, r_i ($0 \le l_i, r_i \le n$) — the cells of the previous and the next element of list for cell i. Value $l_i = 0$ if element in cell i has no previous element in its list. Value $r_i = 0$ if element in cell i has no next element in its list.

It is guaranteed that the input contains the correct description of a single or more doubly linked lists. All lists have linear structure: each element of list except the first has exactly one previous element; each element of list except the last has exactly one next element. Each memory cell contains information about one element from some list, each element of each list written in one of n given cells.

Output

Print n lines, the i-th line must contain two integers l_i and r_i — the cells of the previous and the next element of list for cell i after all lists from the input are united in a single list. If there are many solutions print any of them.

input
7 4 7 5 0 0 0 6 1 0 2 0 4 1 0

output
4 7 5 6 0 5 6 1 3 2 2 4 1 0

P. MAZE

2 seconds, 256 megabytes

A Maze is represented by a 0-1 matrix $a_{N\times M}$ in which $a_{i,j}$ = 1 means cell (i, j) is an obstacle, $a_{i,j}$ = 0 means cell (i, j) is free. From a free cell, we can go up, down, left, or right to an adjacent free cell. Compute the minimal number of steps to escape from a Maze from a given cell (i_0, j_0) within the Maze.

Input

- Line 1 contains N, M, i_0, j_0 $(2 \le N, M \le 900)$
- Line $i + 1$ $(i = 1, \ldots, N)$ contains the i^{th} line of the matrix $a_{N\times M}$

Output

Unique line contains the number minimal of steps to escape the Maze or -1 if no way to escape the Maze.

input
8 12 5 6 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1

output
7