ICPC SOUTH PACIFIC DIVISIONALS LEVEL B

OCTOBER 15, 2022

Contest Problems

A: A Powerful Tower
B: Begrudging Friendship
C: City Planning
D: Dig and Drill
E: Eating Socks
F: Falling Domi ...no, no, NOO!
G: GPU Meme
H: Hopeless Bingo
I : Incredibly Fun Game
J : Jumpy Children
K: Kiwis, Snakes, and Ladders
L: Lugging Ladders

This contest contains twelve problems. Good luck.

For problems that state "*Your answer should have an absolute or relative error of less than* 10^{-9}", your answer, x, will be compared to the correct answer, y. If $|x - y| < 10^{-9}$ or $\frac{|x-y|}{|y|} < 10^{-9}$, then your answer will be considered correct.

Definition 1

For problems that ask for a result modulo m:
If the correct answer to the problem is the integer b, then you should display the unique value a such that:

- $0 \leq a < m$
  and
- $(a - b)$ is a multiple of m.

Definition 2

A string $s_1 s_2 \cdots s_n$ is lexicographically smaller than $t_1 t_2 \cdots t_\ell$ if

- there exists $k \leq \min(n, \ell)$ such that $s_i = t_i$ for all $1 \leq i < k$ and $s_k < t_k$
  or
- $s_i = t_i$ for all $1 \leq i \leq \min(n, \ell)$ and $n < \ell$.

Definition 3

- Uppercase letters are the uppercase English letters ($A, B, \ldots, Z$).
- Lowercase letters are the lowercase English letters ($a, b, \ldots, z$).

Definition 4

Unless otherwise specified, the distance between two points (x_0, y_0) and (x_1, y_1) is defined as its Euclidean distance:

$$\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}.$$

Problem A

A Powerful Tower

Time limit: 2 seconds

Marcell has built a grid of blocks with R rows and C columns. Every block has a lowercase letter written on it.

A contiguous subrectangle of the grid is called a *tower* if it includes all rows of the grid and the letters read left-to-right in each row are the same as the other rows. The *power* of a tower is the number of columns it spans.

What is the maximum power of all towers in the grid?

Input

The first line of input contains two integers, R ($1 \leq R \leq 2\,000$), which is the number of rows, and C ($1 \leq C \leq 2\,000$), which is the number of columns.

The next R lines of input each contain C lowercase letters. The jth letter on the ith line is letter at location (i, j) in the grid of blocks.

Output

If there are no towers, display 0. Otherwise, display the maximum power of all towers in the grid.

Sample Input 1	Sample Output 1
2 3 abc xbc	2

Sample Input 2	Sample Output 2
2 3 abc xcb	0

Sample Input 3	Sample Output 3
1 4 xyfh	4

Sample Input 4	Sample Output 4
3 3 abc efg hij	0

This page is intentionally left (almost) blank.

Problem B
Begrudging Friendship
Time limit: 1 second

Kyle and Leonard have never been the best of friends, but tonight, they are out for a meal together. Kyle has offered to purchase the first round of drinks. All drinks on the menu have different prices.

Kyle would like to order himself a great drink and Leonard a not-so-great drink. But he does not want to be too obvious, so he will not buy himself the most expensive drink and he will not buy Leonard the least expensive drink. He will not buy the same drink for both of them.

If the value of Kyle's drink is k and the value of Leonard's drink is ℓ, then Kyle's *pettiness* value is $k - \ell$. Kyle will choose the two different drinks that maximize the pettiness value without breaking his rule to not buy himself the most expensive drink or buy Leonard the cheapest drink.

Given the prices of all drinks, what is the maximum pettiness value Kyle can achieve?

Input

The first line of input contains a single integer N ($2 \leq N \leq 100$), which is the number of drinks.

The next line contains N distinct integers $d_1, d_2, \ldots, d_N$ ($1 \leq d_i \leq 1\,000\,000$), which are the values of the N drinks.

Output

Display the maximum pettiness value Kyle can achieve.

Sample Input 1	Sample Output 1
4 10 15 5 7	3

Sample Input 2	Sample Output 2
2 10 5	-5

This page is intentionally left (almost) blank.

Problem C
City Planning
Time limit: 5 seconds

Amanda lives in a city with lazy city planners. The city comprises N intersections with roads between them. The roads have a direction; that is, they are one-way. It is possible for a pair of intersections to have roads connecting them in both directions. It is also possible for an intersection to have a road to itself.

The lazy city planners keep a collection of M lists of intersections. The lists are labelled from 1 to M. The roads out of an intersection i are defined by a number ℓ_i, which means that there is an outgoing road from i to each intersection in list number ℓ_i from the collection of M lists.

Amanda lives at intersection 1 and works at intersection N. What is the minimum number of roads she needs to travel down to get from home to work?

Input

The first line of input contains two integers, N ($2 \leq N \leq 500\,000$), which is the number of intersections, and M ($1 \leq M \leq 500\,000$), which is the number of lists.

The next M lines describe the lists. The kth of these lines contains an integer c ($1 \leq c \leq N$) followed by c distinct integers $a_1, a_2, \ldots, a_c$ ($1 \leq a_i \leq N$), which are the intersections in list k. The total length of all lists is at most $500\,000$.

The final line of input contains N integers $\ell_1, \ell_2, \ldots, \ell_N$ ($1 \leq \ell_i \leq M$), which denote that intersection i's outgoing roadways are the list labelled ℓ_i.

Output

Display the minimum number of roads Amanda must travel. If it is impossible for her to reach work, display -1 instead.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
-1
```

This page is intentionally left (almost) blank.

Problem D
Dig and Drill
Time limit: 1 second

Minnie is attempting to dig a very deep hole. Initially, the hole has a depth of 0. She has two tools available to her: a shovel and a drill. The shovel increases the depth of the hole by 1 (that is, if the depth is x prior to using the shovel, the depth is $x + 1$ after using the shovel once). The drill triples the depth of the hole (that is, if the depth is y prior to using the drill, the depth is $3y$ after using the drill once). She may use each tool as many times and in whatever order she wishes.

For example, if she wants to dig a hole of depth 6, she could use the shovel, then the drill, then the shovel again 3 more times (using the tools a total of 5 times). Alternatively, she could have used the shovel twice, then the drill once (using the tools a total of 3 times).

Given the desired depth of the hole Minnie wants to dig, what is the minimum number of times she has to use the tools to dig a hole of that depth?

Input

The input consists of a single line containing a single integer D ($1 \leq D \leq 1\,000\,000$), which is the desired depth.

Output

Display the minimum number of times she has to use the tools.

Sample Input 1	Sample Output 1
6	3

Sample Input 2	Sample Output 2
1	1

Sample Input 3	Sample Output 3
2	2

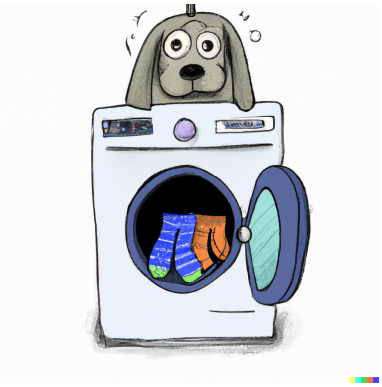
Sample Input 4	Sample Output 4
3	2

Sample Input 5	Sample Output 5
4	3

Sample Input 6	Sample Output 6
500	11

This page is intentionally left (almost) blank.

Problem E
Eating Socks
Time limit: 1 second



ARRRGGG! Not again! I swear that every time I put socks into this dryer, some of them go missing!

I have socks with different colours. Any two socks of the same colour can form a pair. I only ever put pairs of socks into the dryer, but when I pull them out of the dryer, I pull out individual socks and need to put them in pairs again.

For example, if I put 3 pairs of green socks (6 green socks) and 1 pair of blue socks (2 blue socks) into the dryer, but only pulled out 2 green socks and 1 blue sock, then 5 socks went missing (4 green and 1 blue).

I do not remember which colours I put into the dryer, or how many socks, but I definitely put them in as pairs only. Given the colours of the socks I pulled out of the dryer, what is the minimum number of socks that are missing?

Input

The first line of input contains a single integer N ($1 \leq N \leq 100$), which is the number of socks pulled out of the dryer.

The next line contains N integers $c_1, c_2, \ldots, c_N$ ($1 \leq c_i \leq 100$), which are the colours of the socks pulled out of the dryer.

Output

Display the minimum number of socks that are missing.

Sample Input 1	Sample Output 1
3 1 2 1	1

Sample Input 2	Sample Output 2
3 1 2 3	3

Sample Input 3	Sample Output 3
6 1 1 1 1 1 1	0

Sample Input 4	Sample Output 4
5 3 14 15 92 65	5

This page is intentionally left (almost) blank.

Problem F

Falling Domi ...no, no, NOO!

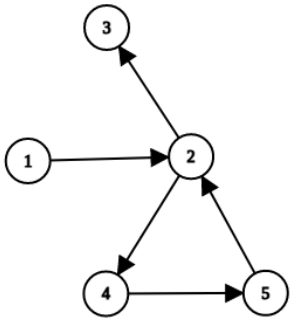
Time limit: 5 seconds

Pauleen has just about finished her picture made out of dominoes. From above, the dominoes form a beautiful portrait of Iggy the Iguana. Unfortunately for Pauleen, dominoes are not very stable.

When one domino is knocked over, it may knock over several other dominoes, then those dominoes may knock over several other dominoes, and so on. Knocking over a single domino can therefore cause many dominoes to be knocked over. Once a domino is knocked over, it cannot be knocked over again.

Consider the setup in the image. If domino 2 is knocked over, it will knock over dominoes 3 and 4 directly, then domino 4 will knock down domino 5. Domino 1 will not be knocked over.

Pauleen has a mischievous brother who likes to ruin her pictures by knocking over just one domino and watching the ensuing chaos. The *chaos factor* for a domino is the total number of dominoes that are knocked over if the brother chooses to knock over that domino. What is the chaos factor for each of the dominoes?



Input

The first line of input contains two integers D ($2 \leq D \leq 100$), which is the number of dominoes, and R ($1 \leq R \leq D(D-1)$), which is the number of domino relationships. The dominoes are numbered 1 to D.

The next R lines describe the domino relationships. Each of these lines contains two integers x ($1 \leq x \leq D$) and y ($1 \leq y \leq D$ and $x \neq y$), which indicate that if domino x is knocked over, then domino y is directly knocked over as a result. No domino relationship will appear more than once.

Output

Display the chaos factor for each of the dominoes in the order of their numbers.

Sample Input 1

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

Sample Output 1

```
5 4 1 4 4
```

Sample Input 2

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

Sample Output 2

```
3 2 1
```

Sample Input 3

```
2 2
1 2
2 1
```

Sample Output 3

```
2 2
```

Sample Input 4

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

Sample Output 4

```
4 4 4 4
```

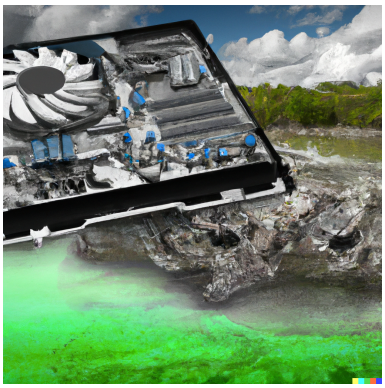
Sample Input 5

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

Sample Output 5

```
4 2 2 1
```

Problem G
GPU Meme
Time limit: 1 second



Varun is very interested in GPUs, which he uses to convert electricity into heat and damage the environment. There are N GPUs on the market, which Varun would like to buy a subset of. Note that these are N individual GPUs and not N types of GPU, so Varun can only buy each at most once.

Each GPU has a monetary cost and an environmental cost. The total environmental costs of all GPUs that Varun buys must not exceed the legal limit. Varun will maximize the total environmental cost of the GPUs he buys, while not exceeding that legal limit. Varun has a lot of money, so can buy all the GPUs if he wants to. However, he is also frugal, so if there are multiple ways to maximize the environmental cost, Varun will choose whichever of those options minimizes the monetary cost.

What is the monetary cost of the subset of GPUs that Varun chooses?

Input

The first line of the input contains two integers N ($1 \leq N \leq 2\,000$), which is the number of GPUs, and E ($1 \leq E \leq 2\,000$), which is the legal limit of environmental cost.

The next N lines describe the GPUs. Each of these lines contains two integers M ($1 \leq M \leq 2\,000$), which is the monetary cost of this GPU, and C ($1 \leq C \leq 2\,000$), which is the environmental cost of this GPU.

Output

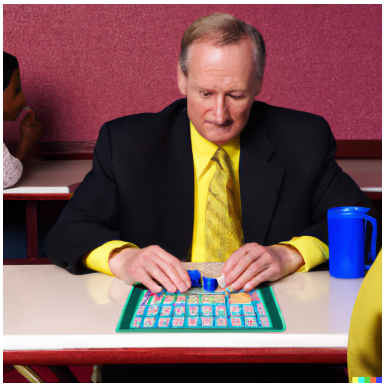
Display the monetary cost of a subset of GPUs that Varun chooses.

Sample Input 1	Sample Output 1
4 10 1 11 1 9 2 1 3 1	3

Sample Input 2	Sample Output 2
3 6 2 3 3 3 1 4	5

This page is intentionally left (almost) blank.

Problem H
Hopeless Bingo
Time limit: 1 second



Bingo is a game of chance for multiple players. Each player gets a card with a (fixed size) subset of the values from 1 to 90. A caller then draws value tokens (also from 1 to 90), calling each out when drawn (traditionally with a nickname, e.g., "66, clickety click"). Each player marks their card's values off when called, and the first player to mark all of their values off is the winner. If multiple players finish marking all their values off at the same time, they are all winners.

Many decades ago, one of your older relatives had a Bingo set but lost some of the tokens used for the random drawing, so any card with those values can never win. When someone commented that this seemed unfair, this was countered with the point that it was a game of chance that included the distribution of the cards, so the game as a whole was fair!

In tonight's game, the remaining tokens will be drawn in a particular order. Given this order, each card's status is either *winning*, *losing*, or *hopeless*. A card is winning if it is (one of) the first of the cards to have all its values drawn. A card is hopeless if it has some value that is lost (that is, not included in the calls at all). A card is losing if it is neither winning nor hopeless.

Given the order of the calls and the contents of all cards, determine the status of each card.

Input

The first line of input contains three integers N ($1 \leq N \leq 89$), which is the number of tokens left in the Bingo set, P ($2 \leq P \leq 10$), which is the number of players, and C ($1 \leq C \leq \min(30, N)$), which is the number of values on each card.

The second line contains the N distinct integers $c_1, c_2, \ldots, c_N$ ($1 \leq c_i \leq 90$), which are the tokens in the order in which they will be drawn (possibly including tokens after a card has won).

The remaining P lines describe the cards. The ith of these lines contains C distinct integers, each in the inclusive range between 1 and 90, which are the values on the ith player's card.

Output

Display the status of each player's card in the order in which they occur in the input.

Sample Input 1

```
5 6 2
2 1 3 90 66
1 3
3 2
4 66
3 90
3 90
90 3
```

Sample Output 1

```
winning
winning
hopeless
losing
losing
losing
```

Sample Input 2

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

Sample Output 2

```
hopeless
hopeless
```

This page is intentionally left (almost) blank.

Problem I
Incredibly Fun Game
Time limit: 1 second

Hopscotch is an incredibly fun game. Several rectangles are drawn on the ground and a player jumps from one to another. Two side-by-side rectangles form a *2-cell* if they have the same height (but may or may not have the same width).

You have a strange can of paint that can paint only an area of exactly P cm^2, no more and no less. You would like to paint the interior of some 2-cell. For example, if you have 4 cm^2 of paint, then you could paint a 2-cell made up of a 1×1 and a 1×3, since the area of these is $1 + 3 = 4$. But this is not the only 2-cell you could paint. You could paint any one of these four 2-cells: $(1 \times 1, 1 \times 3), (1 \times 2, 1 \times 2), (1 \times 3, 1 \times 1), (2 \times 1, 2 \times 1)$. You cannot paint a $(1 \times 1, 1 \times 1)$ since you must use exactly 4 cm^2 of paint.

Given the amount of paint, how many different 2-cells can you paint?

Input

The input consists of a single line containing a single integer P ($2 \leq P \leq 100\,000$), which is the amount of paint you have (in cm^2).

Output

Display the number of different 2-cells you can paint.

Sample Input 1	Sample Output 1
4	4

Sample Input 2	Sample Output 2
2	1

Sample Input 3	Sample Output 3
15	20

This page is intentionally left (almost) blank.

Problem J
Jumpy Children
Time limit: 1 second

In the classic children's game *What time is it, Mr. Wolf?*, several children line up and start walking towards the wolf constantly asking, "what time is it, Mr. Wolf?" When ready, the wolf yells *LUNCH TIME!* and starts chasing the children back to their home base. If the wolf gets to a child before they reach home base, the child is caught. If the child gets to home base before the wolf or they arrive at home base at the same time, the child is not caught. All the children are on a single straight line between the wolf and the home base.

You are the wolf this time. When you yell *LUNCH TIME!* you turn around and see everyone's location. You know how fast you run, as well as the speeds of all of the children, but you do not remember which child runs at each of those speeds.

For example, say there were three children (Alice, Bob, and Charlie) that are 25 m, 35 m, and 40 m from home base, respectively. You (the wolf) are 100 m away from home base and run at 10 m/s. The speeds of the children are 2 m/s, 3 m/s, and 4 m/s, in some order. If Alice goes 2 m/s, Bob goes 3 m/s, and Charlie goes 4 m/s, the wolf will catch Alice and Bob before they get to home base. However, if Alice goes 3 m/s, Bob goes 2 m/s, and Charlie goes 4 m/s, then the wolf will only catch Bob. In this example, no matter how the speeds are assigned, the wolf will always be able to capture at least one child. Note that children may pass each other.

Given the children's locations and the list of speeds, what is the minimum number of children the wolf is guaranteed to capture over all assignments of speeds to children?

Input

The first line of input contains three integers N ($1 \leq N \leq 1\,000$), which is the number of children, D ($2 \leq D \leq 1\,000$), which is the distance that the wolf is from home base, and W ($1 \leq W \leq 1\,000$), which is the speed of the wolf (in m/s).

The next line describes the children's locations. This line contains N integers $\ell_1, \ell_2, \ldots, \ell_N$ ($1 \leq \ell_i < D$), which are the distances the children are from home base (in m).

The next line describes the children's speeds. This line contains N integers $s_1, s_2, \ldots, s_N$ ($1 \leq s_j \leq 1\,000$), which are the speeds of the children (in m/s), in an unspecified order.

Output

Display the minimum number of children the wolf is guaranteed to capture.

Sample Input 1	Sample Output 1
3 100 10 25 35 40 2 3 4	1

Sample Input 2	Sample Output 2
3 100 4 35 40 25 2 3 4	0

Sample Input 3	Sample Output 3
2 2 100 1 1 100 1	1

Sample Input 4	Sample Output 4
3 2 100 1 1 1 5 10 15	3

Problem K
Kiwis, Snakes, and Ladders
Time limit: 1 second

Snakes and Ladders is Janet's favourite classic board game. There are 100 cells on the 10×10 board, which are numbered 1 to 100. The cells are in the following configuration:

```
100  99  98  97  96  95  94  93  92  91
 81  82  83  84  85  86  87  88  89  90
 80  79  78  77  76  75  74  73  72  71
 61  62  63  64  65  66  67  68  69  70
 60  59  58  57  56  55  54  53  52  51
 41  42  43  44  45  46  47  48  49  50
 40  39  38  37  36  35  34  33  32  31
 21  22  23  24  25  26  27  28  29  30
 20  19  18  17  16  15  14  13  12  11
  1   2   3   4   5   6   7   8   9  10
```

Janet has played the game so many times that she has introduced a variation: Kiwis, Snakes, and Ladders.

There are four players playing the game. All four players start on cell 1. The players play in turns (player 1, then 2, then 3, then 4, then 1, then 2, and so on). A game uses a deck containing N cards, each with a number between 1 and 6 inclusive written on it. On a player's turn, they draw a card and move forward that many cells. If that card makes them go past cell 100, then they are finished and no longer take turns in the future. Otherwise, the player looks at the cell they landed on. Each cell contains one of a snake, a ladder, a kiwi, or nothing. If there is nothing in the cell they landed on, their turn is over. If there is a snake or ladder, they are teleported to the cell corresponding to that snake or ladder and then their turn is over (the contents of the new cell they teleported to are ignored). If there is a kiwi, they remain on that cell but miss their next turn (they do not draw a card nor move on a missed turn).

The game is over after a total of N turns or after all players have finished (whichever comes first). Given the board and the cards in the deck, where are each of the four players when the game is over?

Input

The first 10 lines of input describe the 10×10 board. Each of these lines contains 10 integers describing the 10 cells in this row of the board. Each cell is represented by an integer b ($-1 \leq b \leq 100$). If $b = -1$, then the cell is a kiwi, if $b = 0$, then the cell is empty. Otherwise, if $1 \leq b \leq 100$, then this is a snake/ladder that teleports you to cell number b.

The next line contains a single integer N ($1 \leq N \leq 100$), which is the number of cards in the deck.

The next line contains N integers $c_1, c_2, \ldots, c_N$ ($1 \leq c_i \leq 6$), which are the cards in the deck, in order.

Output

Display the location of each player at the end of the game, in order. If a player has finished the game, their location is 101.

Sample Input 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 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 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 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
9
6 6 6 6 6 6 6 6 6
```

Sample Output 1

```
19 13 13 13
```

Sample Input 2

```
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 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 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 0 0
0 50 0 0 0 0 0 0 0 0
0 100 0 0 0 0 0 0 0 0
9
1 6 6 6 6 6 6 6 6
```

Sample Output 2

```
101 50 13 13
```

Sample Input 3

```
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 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 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 0 0
0 0 0 0 0 0 0 0 0 0
0 2 3 4 5 6 7 8 9 10
1
1
```

Sample Output 3

```
2 1 1 1
```

Sample Input 4

```
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 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 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 0 0
0 0 0 0 0 0 0 0 0 0
0 100 0 0 0 0 0 0 0 0
9
1 1 1 1 2 2 2 2 3
```

Sample Output 4

```
101 101 101 101
```

Sample Input 5

```
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 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 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 0 0
0 0 0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0 0 0
11
1 2 2 2 3 3 3 4 4 4 4
```

Sample Output 5

```
6 10 10 10
```

This page is intentionally left (almost) blank.

Problem L
Lugging Ladders
Time limit: 1 second

Bob recently remarked that they felt as though they were in a "How many ... to change a lightbulb?" joke, as they were finding it hard to get their ladder down the stairs to change a lightbulb and then back up the stairs again afterward. Bob has suitable places to store the ladder both upstairs and downstairs.

Bob has thought of four strategies for storing the ladder. The *efficiency* of a strategy is the number of times the ladder moved to a different level (either from downstairs to upstairs or from upstairs to downstairs).

Determine the efficiency of these four strategies:

1. Always store the ladder upstairs.
2. Always store the ladder downstairs.
3. Leave the ladder on the level it was last used (with the ladder starting the first day upstairs).
4. Leave the ladder on the level it was last used (with the ladder starting the first day downstairs).

Input

The input consists of a single line containing a single string. The length of the string is between 1 and 100, inclusive. The letters in the string represent the sequence of lightbulb change locations, using `U` for upstairs and `D` for downstairs.

Output

Display the efficiency of the four strategies.

Sample Input 1	Sample Output 1
`UUUDD`	`4 6 1 2`

This page is intentionally left (almost) blank.