Spring Coding Bowl ’21 P1 – Counterfeit Detection

Time Limit: 2.0s **Memory Limit:** 256M

In your strange local currency, there should only be $\$4$, $\$6$ and $\$25$ coins. Unfortunately, a counterfeiter just added a whole bunch of fake $\$2$ coins into circulation!

Your job is to determine how many counterfeited coins are mixed into a row of coins. This is more difficult than it looks. The coins are rectangular, so a row of coins looks something like this:

In order to count all the coins, you use a scanning machine that reads the digits on the top of the coins one by one. For the row of coins shown above, your machine will produce the string `622544252`.

Given a sequence of digits generated by the machine, please determine how many of the coins are counterfeit $\$2$ coins. As there are no $\$5$ coins in circulation, you can assume that if you see `25` in the sequence, it represents a non-counterfeit $\$25$ coin. Otherwise, if you see a `2` in the sequence that is not followed by a `5`, you can assume that it is a counterfeit coin.

Input Specification

The first and only line of input will contain a sequence of digits from your coin-scanning machine, such as `622544252`.

Output Specification

Please output the number of counterfeit ($\$2$) coins in the row of coins.

Constraints and Partial Marks

For all test cases, the string is 999 characters or fewer in length.

Additionally, for 4 out of 10 available marks, there are no $\$25$ coins, so the string doesn't contain the digit `5`.

Sample Input

```
2256624425252
```

Sample Output

```
3
```

Explanation for Sample Output

The sample input represents this row of coins:

2	25	6	6	2	4	4	25	25	2

In this row, there are three counterfeit $\$2$ coins.

Spring Coding Bowl '21 P2 – Emerald Exchange

Time Limit: 2.0s **Memory Limit:** 256M

You are walking along the street when a street vendor offers you a special deal. He has a row of N emeralds with different sizes. Each emerald has an integer weight between 1 and 100, inclusive, and the weights are clearly labelled for each emerald. However, the vendor tells you that you should not touch or buy the emeralds with odd-numbered weights as they have dangerous magical properties.

The vendor offers you to buy any number of consecutive emeralds from his row, as long as that selection of emeralds does not include any odd-weighted emeralds.

What is the largest total weight of emeralds you can buy with one purchase, following the rule above?

Input Specification

The input will consist of two lines. The first line contains N, the number of emeralds in the vendor's row. The second line contains N space-separated integers between 1 and 100 inclusive, the weight of each emerald in the order that the vendor has them on display.

Output Specification

Please output one integer: the maximum possible sum of the weights of the emeralds that can be purchased as described above.

Constraints and Partial Marks

For 5 of the 10 available marks, $1 \leq N \leq 3000$.

For the remaining 5 marks, $1 \leq N \leq 10^5$.

Sample Input

```
13
2 3 4 4 5 6 1 2 2 2 1 8 2
```

Sample Output

```
10
```

Explanation of Sample Output

Buying the two rightmost emeralds, with sizes 8 and 2 (sum = 10), is the optimal purchase.

There are many suboptimal alternative purchases. These include:

- Buying the three consecutive emeralds with size 2 (sum = 6),
- Buying two of the three consecutive emeralds with size 2 (sum = 4),
- Buying the two consecutive emeralds with size 4 (sum = 8), or
- Buying a single emerald of size 2, 4, 6, or 8.

Spring Coding Bowl ’21 P3 – Long Pizza

Time Limit: 2.0s **Memory Limit:** 256M

You are in charge of adding cheese topping to a pizza prepared for a special customer. This pizza is a very long rectangular strip. It is divided into N slices, numbered 1 to N from left to right. To add cheese topping to the pizza, you have a special machine that can add 1 unit of cheese on each slice in a consecutive group of slices (for example, you can add 1 unit of cheese on each slice between slice 2 and 4 inclusive, adding a total of 3 units of cheese). Each slice can carry an unlimited amount of cheese.

You plan to run the machine R times, each time adding 1 unit of cheese to each slice in a given range.

The customer is on a diet. When he eats the section of pizza from slice x to slice y, inclusive, he would like to know how many units of cheese he is consuming.

NOTE FOR PYTHON USERS: If your program receives TLE (time limit exceeded), you should try submitting using the PyPy interpreter: when you are submitting your code, try using "PyPy 3" or "PyPy 2" as the language, instead of "Python 3" or "Python 2".

Input Specification

The first line will contain the integer N, the length of your long pizza.

The second line will consist of two space-separated integers x and y, indicating that the customer is planning to eat every slice between slice x and slice y inclusive.

The third line will contain the integer R, the number of times you plan to run the topping machine.

The following R lines will each describe one planned run of the topping machine using 2 space-separated integers, l and r $(l \leq r)$, indicating that the machine will add 1 unit of cheese onto each slice between slices l and r, inclusive.

Output Specification

Please output the total number of units of cheese on all of the slices the customer is planning to eat.

Constraints and Partial Marks

For 4 of the 10 available marks, $R \leq 1000$ and $N \leq 10^5$.

For the remaining 6 marks, $R \leq 4000$ and $N \leq 10^7$.

Sample Input

```
10
3 5
3
2 6
4 5
3 3
```

Sample Output

```
6
```

Explanation of Sample Output

After running the machine three times, the amount of cheese on each slice of pizza is as follows:

```
Slice #:  1  2  3  4  5  6  7  8  9  10
Cheese:   0  1  2  2  2  1  0  0  0  0
```

Therefore, slices $3 - 5$ have a total of 6 units of cheese.

Spring Coding Bowl '21 P4 – Trampoline Jump

Time Limit: 2.0s **Memory Limit:** 256M

The Fibonacci Sequence is a mathematical sequence where the first and second terms are 1, and each term after that is the sum of the two previous terms. More formally, if f_i denotes the ith term of the Fibonacci Sequence, then $f_1 = f_2 = 1$, and $f_k = f_{(k-1)} + f_{(k-2)}$ for all integer $k > 2$.

The modulo (or mod) operation is an operation represented by the $\%$ symbol. The expression $a \% b$ denotes the remainder when the positive integer a is divided by the positive integer b. For example, $7 \% 2 = 1$, because the remainder when 7 is divided by 2 is 1. The modulo operator is available in most programming languages using the operator `%` - for example, the line `int a = 7%2;` in C++ will set `a` to be 1.

You have found yourself at the first house of a long street. There are a total of N houses along this street, numbered 1 to N. You are trying to travel from the first house to the last house as fast as possible.

You can always walk from one house to an adjacent house (i.e. from house w to house $w-1$ or house $w+1$, if they exist). This takes one minute.

At each house, there is also a trampoline that allows you to travel a larger distance. In particular, at the ith house, there is a trampoline with strength a_i. Using the trampoline also takes one minute, and from house i, you can use the trampoline to jump either backwards to house $i - a_i$ (as long as $i - a_i \geq 1$) or forwards to house $i + a_i$ (as long as $i + a_i \leq N$).

You have been able to reverse-engineer a pattern of the values of a_i for all the houses:

$a_i = (f_i \% 2021) + (i \% 50)$ for all $1 \leq i \leq N$,

where f_i represents the ith term in the Fibonacci Sequence and $\%$ represents the modulo operation, as defined at the beginning of the problem.

For reference purposes, $a_1 = 2$, $a_2 = 3$, $a_3 = 5$, $a_4 = 7$, $a_5 = 10$, $a_6 = 14$, $a_7 = 20$, and $a_{125} = 356$.

Please find the **minimum** number of minutes you need to travel from house 1 to house N.

NOTE FOR PYTHON USERS: If your program receives TLE (time limit exceeded), you should try submitting using the PyPy interpreter: when you are submitting your code, try using "PyPy 3" or "PyPy 2" as the language, instead of "Python 3" or "Python 2".

Input Specification

The input will consist of one integer, N, the number of houses on the street.

Output Specification

Please output the minimum number of minutes you need to travel from house 1 to house N.

Constraints and Partial Marks

For 3 of the 10 available marks, $N \leq 400$.

For the remaining 7 marks, $N \leq 5 \cdot 10^5$.

Sample Input 1

```
9
```

Sample Output 1

```
3
```

Explanation For Sample Output 1

The first 9 terms of the sequence a_i are $2, 3, 5, 7, 10, 14, 20, 29,$ and 43. To travel from house 1 to house 9, the optimal pattern of moves is:

- Move 1: Jump from house 1 to house 3. This is possible as $1 + a_1 = 1 + 2 = 3$.
- Move 2: Jump from house 3 to house 8. This is possible as $3 + a_3 = 3 + 5 = 8$.
- Move 3: Walk from house 8 to house 9. This is possible as $8 + 1 = 9$.

Every move takes 1 minute, so the output is `3`.

Sample Input 2

```
27
```

Sample Output 2

```
4
```

Explanation For Sample Output 2

One way to reach house 27 in 4 moves is as follows:

- Move 1: Jump from house 1 to house 3. This is possible as $1 + a_1 = 1 + 2 = 3$.
- Move 2: Jump from house 3 to house 8. This is possible as $3 + a_3 = 3 + 5 = 8$.
- Move 3: Walk backwards from house 8 to house 7. This is possible as $8 - 1 = 7$.
- Move 4: Jump from house 7 to house 27. This is possible as $7 + a_7 = 7 + 20 = 27$.

Spring Coding Bowl '21 P5 – Woodcutting Game

Time Limit: 2.0s **Memory Limit:** 256M

After tearing down your pig barn, you and your friend Tyler have to dispose of two long rectangular wooden boards. You have to cut the boards into smaller pieces. Because this is boring, you devise a game to play to make things more interesting.

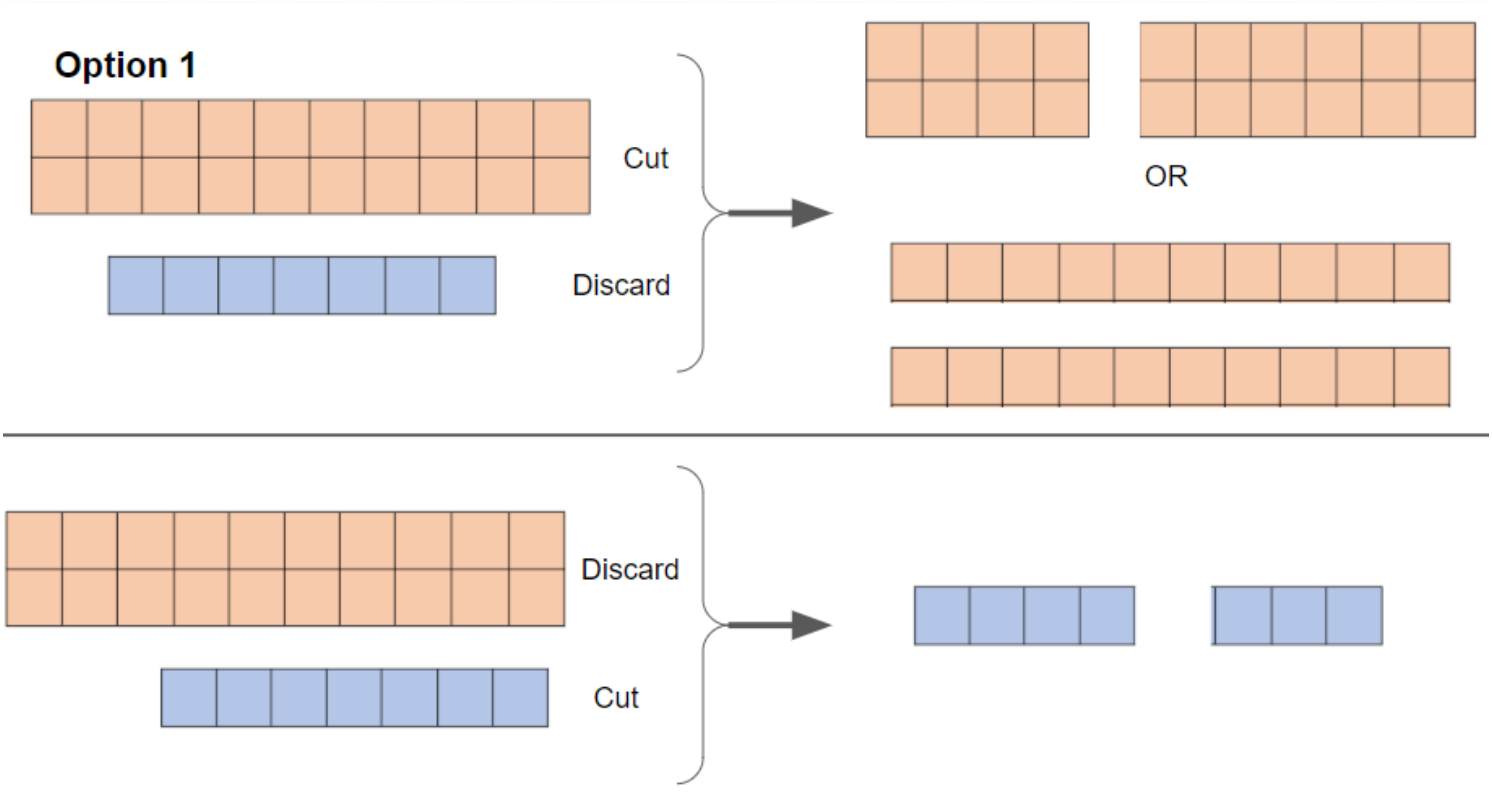
The two 2-dimensional rectangular boards have **heights of either 1 or 2** and integer widths. For example, one board can be a 2×2021 board, and another can be a 1×5 board.

You and Tyler will take turn cutting the rectangular boards, and you will go first. Each of the cuts you make must turn one rectangular board into two smaller rectangular boards, each with an integer height and width. The cuts must be made either horizontally or vertically.

In particular, when it is your turn, you have two options as follows. Note that both options leave your opponent with two rectangles, so the game can continue.

Option 1: Choose one board to discard entirely, and then cut the other board into two smaller integer-dimension rectangles however you want.

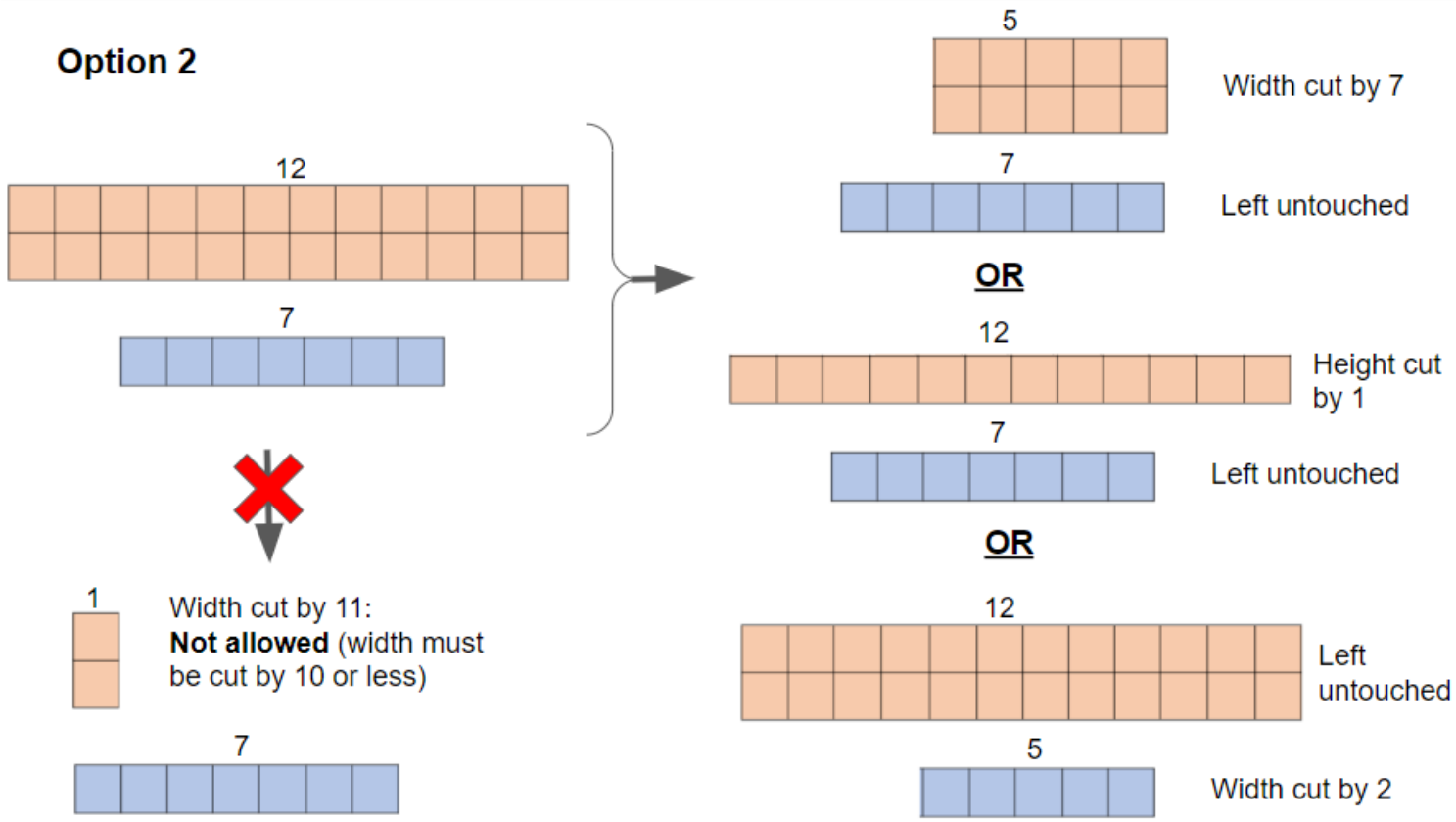
The below diagram illustrates this option (but does not include all possible ways to cut the rectangle).



Option 2: Choose one board to leave untouched, cut a rectangular piece off the other board, and discard that piece.

If you choose Option 2, you must **EITHER cut 1 off the height OR cut any integer between 1 and 10 (inclusive) off the width** of the board that you have chosen to cut.

The below diagram illustrates this option (but does not include all possible ways to cut the rectangle).



You move first. The player who leaves the other player with two 1 by 1 boards that can no longer be cut further is declared the winner.

You are deciding whether to place a bet against Tyler, so you want to know **whether it is possible for you to guarantee a win**, given the starting dimensions of the two boards.

NOTE FOR PYTHON USERS: If your program receives TLE (time limit exceeded), you should try submitting using the PyPy interpreter: when you are submitting your code, try using "PyPy 3" or "PyPy 2" as the language, instead of "Python 3" or "Python 2".

Input Specification

The input will be one line, containing four space-separated integers: $h_1, w_1, h_2,$ and w_2, indicating that the two boards you start with are $h_1 \times w_1$ and $h_2 \times w_2$.

Output Specification

Output the character `W` if you can force a win, or `L` if your opponent Tyler can force a win.

Constraints and Partial Marks

For 2 of the 10 available marks, $h_1 = h_2 = 1$ and $w_1, w_2 \leq 10$.

For another 2 of the 10 available marks, $h_1 = h_2 = 1$ and $w_1, w_2 \leq 300$.

For another 2 of the 10 available marks, $1 \leq h_1, h_2 \leq 2$ and $w_1, w_2 \leq 10$.

For the remaining 4 marks, $1 \leq h_1, h_2 \leq 2$ and $w_1, w_2 \leq 300$.

Sample Input 1

```
2 2 1 3
```

Sample Output 1

```
W
```

Explanation for Sample Output 1

The sample input indicates that you start with a 2×2 board and a 1×3 board.

On your first turn, if you ignore the 2×2 board, and cut 2 off the width of the 1×3 board to leave a 1×1 board (this is permitted in Option 2), Tyler will get a 2×2 board and a 1×1 board on his turn.

It is easy to see that no matter how Tyler plays his move, he will leave you with at least one 1×2 board. As such, on your second turn, you just need to cut a 1×2 board into two 1×1 boards, and discard the other board (this is allowed in Option 1). Now, Tyler is left with two 1×1 boards that cannot be cut further, so he loses, and you win.

Since you are able to force a win, the output is `W`.

Sample Input 2

```
2 2 1 1
```

Sample Output 2

```
L
```

Explanation for Sample Output 2

Similar to the example above, if you start with a 2×2 board and a 1×1 board, no matter what you do, you will always leave Tyler with at least one 1×2 board. Tyler can use Option 1 and cut the 1×2 board into two 1×1 boards and discard the other board, which is a loss for you.

Since you cannot force a win (instead, Tyler can force you to lose), the output is `L`.