A Frobenius

Problem

The *Frobenius problem* is an old problem in mathematics, named after the German mathematician G. Frobenius (1849–1917).

Let $a_1, a_2, \ldots, a_n$ be integers larger than 1, with greatest common divisor (gcd) 1. Then it is known that there are finitely many integers larger than or equal to 0, that cannot be expressed as a lineair combination $w_1a_1 + w_2a_2 + \ldots + w_na_n$, using integer coefficients $w_i \geq 0$. The largest of such nonnegative integers is known as the *Frobenius number* of $a_1, a_2, \ldots, a_n$ (denoted by $\mathrm{F}(a_1, a_2, \ldots, a_n)$). So: $\mathrm{F}(a_1, a_2, \ldots, a_n)$ is the largest nonnegative integer that cannot be expressed as a nonnegative integer linear combination of $a_1, a_2, \ldots, a_n$.

For $n = 2$ there is a simple formula for $F(a_1, a_2)$. However, for $n \geq 3$ it is much more complicated. For $n = 3$ only for some special choices of a_1, a_2, a_3 formulas exist. For $n \geq 4$ no formulas are known at all.

We will consider here the Frobenius problem for $n = 4$. In this case our version of the problem can be formulated as follows. Let four integers a, b, c and d be given, with $a, b, c, d > 1$ and $\gcd(a, b, c, d) = 1$. We want to know two things.

- How many nonnegative integers less than or equal to $1,000,000$ cannot be expressed as a nonnegative integer linear combination of the values a, b, c and d ?
- Is the Frobenius number of a, b, c and d less than or equal to 1,000,000 and if so, what is its value?

Input

The input consists of a single test case, satisfying the following format:

- One line, containing four integers a, b, c, d (with $1 < a, b, c, d \leq 10,000$ and $\gcd(a, b, c, d) = 1$), separated by single spaces.

Output

The output should contain two lines.

- The first line contains the number of integers between 0 and 1,000,000 (boundaries included) that cannot be expressed as $a \cdot w + b \cdot x + c \cdot y + d \cdot z$, where w, x, y, z are nonnegative (meaning ≥ 0) integers.
- The second line contains the Frobenius number if this is less than or equal to 1,000,000 and otherwise -1, meaning that the Frobenius number of a, b, c and d is larger than 1,000,000.

Sample Input 1

```
8 5 9 7
```

Sample Output 1

```
6
11
```

Sample Input 2

```
5 8 5 5
```

Sample Output 2

```
14
27
```

Sample Input 3

```
1938 1939 1940 1937
```

Sample Output 3

```
600366
-1
```

B The Great Cleanup

Problem

It happens to all of us. While you are happily downloading a movie or copying a file, a warning appears on your computer screen: "disk full," or "disk quota exceeded."
There are several ways to deal with this. You may simply accept the fact that you will never watch the movie you were downloading, or that you have to live the rest of your life with a single copy of the file. You may also install a larger disk or try to acquire a larger disk quota.
A third option is to create some space on the disk by removing files that you do not really need anymore. Under Linux, you can use the command `rm` for this. The syntax is simple: `"rm filename"` removes the file with name `"filename"`. You may use a separate rm-command for every single file, but you may also use a wildcard `'*'` to remove multiple files in one step. For example, `"rm BAPC*"` removes all files that start with `"BAPC"`. This way, you have to type fewer commands.
Of course, you must not remove files that you want to keep. Hence, `"rm *"`, which is allowed, and which removes all files in the current directory, is often not desirable.
Now, given the names of the files you want to remove, and given the names of the files you want to keep, you have to determine the minimum number of rm-commands to get the job done. You may only use wildcards *at the end* of your commands. For example, `"rm *.txt"` (which would remove all `".txt"` files) is not allowed.

Input

The input consists of a single test case, satisfying the following format:

- One line with an integer N_1, satisfying $1 \leq N_1 \leq 1000$: the number of files that must be removed.
- N_1 lines, each with the name of one file that must be removed.
- One line with an integer N_2, satisfying $0 \leq N_2 \leq 1000$: the number of files that must not be removed.
- N_2 lines, each with the name of one file that must not be removed.

Each filename is a string x with $1 \leq \mathit{Length}(x) \leq 20$, consisting of alphabetic characters (upper case and lower case), digits and/or periods. That is, characters from the set { A, B, C, ..., Z, a, b, c, ..., z, 0, 1, 2, ..., 9, . }. All $N_1 + N_2$ file names for a test case are different.

Output

The output should contain a single number, on a single line: the smallest number of rm-commands to remove exactly the right files.

Sample Input 1

```
11
BAPC.in
BAPC.out
BAPC.tex
filter
filename
filenames
clean
cleanup.IN1
cleanup.IN2
cleanup.out
problem.tex
5
BAPC
files
cleanup
cleanup.IN
cleaning
```

Sample Output 1

```
8
```

C Have a Nice Day

Problem

Rumour has it that the $\mathcal{P}$ versus $\mathcal{NP}$ question has been solved: the two classes are not equal. This implies that many well-known problems, such as the Traveling Salesman Problem, will remain difficult forever. It can be considered a waste of time to search for polynomial time solutions: essentially only brute-force approaches can solve them. Nothing you can do about that.
In view of the international crisis, the new Dutch government has therefore announced that on certain days it is not allowed to work on these hard problems anymore. Instead, one must concentrate on easier issues. These days are called *nice*. Of course, the algorithm to decide whether a given date is nice or not should itself be easy. So far politicians could not find such an algorithm. Can you?
A date *day month year* is written down using the digits 0,...,9. A date is called *nice* if the digits occurring in it occur an equal number of times, *and* if it can be *split*. A date can be split if its *four number set* can be divided into two disjoint subsets with equal sum; the four numbers are the day, the month, the left part of the year (the number represented by its first and second digit; for 1957 this is 19) and the right part of the year (the number represented by its third and fourth digit; for 2000 this is 0). For example, 16 5 4928 is nice, because all digits occur exactly once and $16 + 5 + 28 = 49$.

Input

The input consists of a single test case, satisfying the following format:

- One line with three integers D, M and Y separated by single spaces, satisfying $1 \leq D \leq 31$, $1 \leq M \leq 12$ and $1000 \leq Y \leq 9999$: the day, month and year of a valid date, respectively. There are no leading zeros; e.g., June is represented as 6 and not as 06.

Output

The output should contain the string `"yes"` or `"no"`: the fact whether the date is nice or not.

Sample Input 1

```
16 5 4928
```

Sample Output 1

```
yes
```

Sample Input 2

```
14 12 2747
```

Sample Output 2

```
no
```

Sample Input 3

```
11 11 1111
```

Sample Output 3

```
yes
```

Sample Input 4

```
3 3 2014
```

Sample Output 4

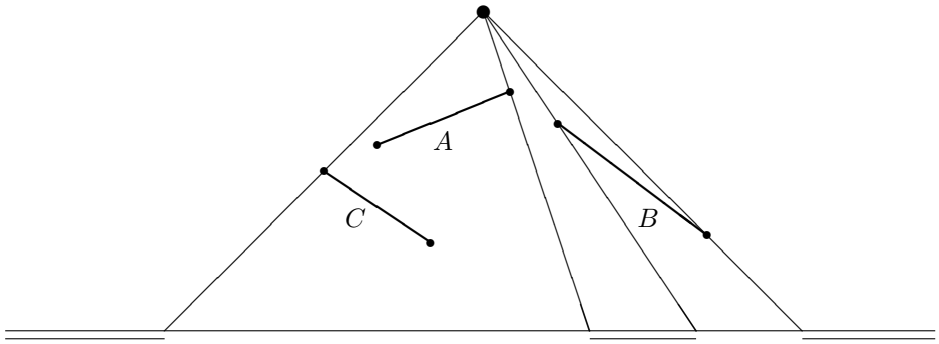
```
no
```

This page intentionally left blank.

D Lucky Light

Problem

We have a (point) light source at position (x_L, y_L) with $y_L > 0$, and a finite series of line segments, all of finite non-zero length, given by the coordinates of their two endpoints. These endpoints are all different. The line segments are all situated above the x-axis ($y = 0$).
The segments cast their shadows onto the x-axis. We assume that the shadows of two segments either do not overlap at all, or have an overlap that has some non-zero width (they do not just touch). We also assume that for each segment its shadow is more than just one point, i.e., there is no segment that is directed toward the light source. The height of the light source (y_L) is at least 1 unit larger than the y-coordinates of the endpoints of the line segments. This guarantees that indeed each line segment has a bounded shadow on the x-axis.
The collection of shadows divides the x-axis into dark and lighted areas (intervals). The problem is to determine the number of lighted areas — which is at least 2 (if there is at least one line segment, otherwise it is 1).
In the picture below the three line segments A, B and C cause three lighted areas, as indicated.



Input

The input consists of a single test case, satisfying the following format:

- One line with one integer n with $0 \leq n \leq 100$: the number of line segments.
- One line with two integers x_L and y_L, the coordinates of the light source, separated by a single space. The coordinates satisfy $-100 \leq x_L \leq 100$ and $1 < y_L \leq 1,000$.
- n lines, each containing four integers x_i, y_i, u_i and v_i, separated by single spaces, that specify x- and y-coordinates of the two endpoints (x_i, y_i) and (u_i, v_i) of the i^{th} line segment, where $-100 \leq x_i, u_i \leq 100$ and $0 < y_i, v_i < y_L$, for $1 \leq i \leq n$.

Output

The output should contain a single number, on a single line: the number of lighted areas.

Sample Input 1

```
3
50 60
55 45 30 35
64 39 92 18
20 30 40 16
```

Sample Output 1

```
3
```

This test case corresponds to the picture in the problem description.

Sample Input 2

```
2
-10 50
-10 1 10 11
-10 11 10 1
```

Sample Output 2

```
2
```

This test case has two crossing line segments.

E Stock Market

Problem

The bankers of PigsBank, Inc. have collected day-to-day data of the daily profits (and losses) of the shares they hold. Based on these numbers they decide to calculate which days would have been the most profitable to buy and sell their shares, so they can compare that with their actual gain.
Your task is to write a program that computes the optimal 'run' in the sequence of numbers, the subsequence that maximizes the profit. The subsequence you compute is represented by the indexes of the first and last numbers in the subsequence, where we start counting indexes by 1: PigsBank, Inc. bankers are not C programmers. We are asking for exactly one date to buy and one date to sell shares since otherwise the solution would be simple: keep your shares on dates that the profit is non-negative.
Unfortunately that will not help PigsBank, Inc. to avoid the losses they had in the past. Instead they can use it to show future customers how profitable the market could have been if they had been taking the right decisions.

Input

The input consists of a single test case, satisfying the following format:

- One line with a single integer N, satisfying $1 \leq N \leq 1,000,000$: the length of the sequence to follow.
- One line with N integers p_i, satisfying $-1,000 \leq p_i \leq 1,000$: the profit (or loss) on date i. The integers are separated by single spaces. At least one of the integers is positive.

Output

The output should contain a single line with two integers i and j, satisfying $1 \leq i \leq j \leq N$, such that the sum of the i-th until the j-th integer is maximized, boundaries included. When i and j have different solutions, choose minimal i and minimal j.

Sample Input 1

```
11
-3 1 -1 2 3 1 -1 2 -3 -5 7
```

Sample Output 1

```
2 8
```

Sample Input 2

```
9
1 -2 3 -1 -1 3 -2 2 -4
```

Sample Output 2

```
3 6
```

This page intentionally left blank.