CIMS Printer

There is only one printer working at Courant Institute of Mathematical Sciences. There are many printing jobs queuing for that printer and people wait for a long time to get their printouts. Each printing job is assigned a priority between 1 (lowest) and 9 (highest). The printing system manages the jobs as follows.

- Select the first job from the queue.
- If there is some job in the queue with higher priority than the selected job then move the selected job to the end of the queue without printing it.
- Otherwise, print that job.

Your task is to write a program to figure out when a given job is printed. The program is given the current printing queue and the position of a job of interest in the queue. It should output how long it will take before that job is printed. Assume that

- there are no additional print jobs added to the queue,
- printing a single job takes exactly one minute, and
- the queue operations happen instantly (i.e., there is no additional time spent to manage the queue).

Input

The first line of input contains two integers `N` and `M` where `N` stands for the number of jobs in the print queue (`1 <= N <= 100`) and `M` is the position of the job (`0 <= M <= N - 1`; with 0 being the first position in the queue). The second line has `N` integers in the range, giving the priorities of jobs in the queue from the first position to the last.

Output

You should print **one line** with a single number, the number of minutes until the job is printed.

Example 1

```
Input:
1 0
5

Output:
1
```

There is only one job and the printing is completed in one minute.

Example 2

```
Input:
4 2
1 2 3 4

Output:
2
```

The job of interest is the job with priority 3.

Queue operations: The first job has priority 1, so it is moved to the back of the queue (since there are jobs with higher priority in the queue). The next job has priority 2, so it is moved to the back of the queue. The next job has priority 3 and it is moved to the back of the queue. Finally, the next job has priority 4 and it will be printed.

Printing takes 1 minute.\ Queue operations: The first job has priority 1, so it is moved to the back of the queue. The next job has priority 2, so it is moved to the back of the queue. Finally, the next job has priority 3 (this is the job of interest) and it will be printed.

Printing takes 1 minute.

In total, two minutes have passed for the job to be printed.

Example 3

```
Input:
6 0
1 1 9 1 1 1

Output:
5
```

Median Rainfall

With the changing climate patterns the researchers are interested in comparing the median amounts of rainfall to the historical data. As a junior developer on the team, you are tasked with writing a program that, for each new hourly rainfall amount generates a new median rainfall amount. Each hour a new data entry comes in with the amount of rain for that hour. The program needs to produce a new median since it might have changed from the last record.

If the current number of hourly records is odd, then the median is the middle element when the values are in sorted order. For example, for the sequence {1, 3, 6, 2, 7}, the median is 3.

If the current number of hourly records is even, then the median is the average of the two middle elements when the values are in sorted order. For example, for the sequence {1, 3, 6, 2, 7, 8}, the median is $(3+6)/2 = 4$ (note that the last operation uses integer division).

Input

The input consists of series of rainfall measurements `R` (`0 <= R <= 2^31`). The total number of hourly measurements `N` is less than 100,000. Each line contains only a single integer, but the numbers may have leading or trailing spaces.

Output

For each input line (each new rainfall measrument), print the current value of the median rainfall amount.

Example 1

```
Input:
1
3
4
60
70
50
2

Output:
1
2
3
3
4
27
4
```

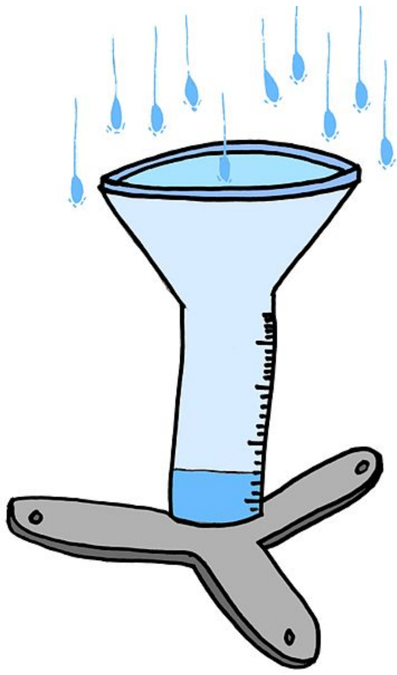

Figure 1: Cleonard1973, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Time Master

Eto manages all timers in the Light Kingdom. People usually need her to repeatedly alert them at a fixed interval. They can register a reminder with her using the following instruction:

Register *id* *interval*

where *id* is the ID of that reminder and *interval* is the interval between two consecutive alerts. Eto will first alert the client after *interval* number hours of registering request and then alert them every *interval* hours.

There are a bunch of different reminder-requests currently on file. Your task is to process the first N reminders for Eto. If there are more than one alerts occurring at the same time, you should list them in the ascending order of *id*.

Input

The input consists of two parts. Each line of the first part contains a single request to register a reminder in the format described above where 1 <= #Requests <= 1000, 1 <= *id* <= 3000 and 1 <= *interval* <= 3000. There are no duplicate *id*'s.

The first part is followed by a single line containing `#` character and a line containing a single integer N (1 <= N <= 10000), representing the number of alerts you need to process for Eto.

Output

You should print the *id*'s of the first N alerts. One *id* per line.

Example 1

```
Input:
Register 2004 200
Register 2005 300
#
5

Output:
2004
2005
2004
2004
2005
```

Four-Prime Numbers

A number p is a prime factor of a number n if p divides n and p is prime. For example, 2, 3, and 5 are prime factors of 30 and 60. A number n has exactly m distinct prime factors if each of p1, p2, . . . pm is a prime factor of n, and n has no other factors, and no two factors pj and pk are equal. For example 12 has exactly two prime factors, 2 and 3, since $12 = 2 * 2 * 3$ and 24 also has two prime factors, since $24 = 2 * 2 * 2 * 3$. Each prime is allowed to occur more than once in the factorization.

A number `C` is *four-prime number* if C is the product of exactly **four distinct prime numbers**. For example, 210 is a four-prime number, since $210 = 2 * 3 * 5 * 7$ (in fact, it is the smallest four prime number), but 16 is not a four-prime number since it has four factors of 2.

Given a positive integer M, find how many values smaller than or equal to M are four-prime numbers.

Input

Input is a single line containing a number 6 <= M <= 5,000,000.

Output

A single line containing the count of the four-prime numbers smaller than or equal to M.

Example 1

```
Input:
210

Output:
1
```

Since 210 is the only four-prime number smaller than or equal to 210 ($210 = 2 * 3 * 5 * 7$).

Example 2

```
Input:
330

Output:
2
```

Since 330 is four-prime number with factors 2, 3, 5, and 11, and 210 is a four-prime number.

Example 3

```
Input:
500

Output:
4
```

These are 210, 330, 390 and 462.

Sia's Box

Sia has a mysterious box and she wants to play a game with you. The box supports two types of operations:

- `1 x`: Put the number x into the box.
- `2`: Take out a number from the box.

Sia will give you a sequence of operations and the results of the above `2` operations. Your task is to determine what is really hidden in the box: a stack, a queue, a max priority queue or something else.

Input

The first line of the input contains a single integer N (0 <= N <= 1000), indicating the number of operations. Each of the next N lines contains a single operation described above. For operation `2`, there is an additional number x indicating the result of that operation. The value of x in both operations satisfies 1 <= x <= 100.

Output

You should print one of the following answers on a line by itself:

- `stack` if it is certain that the box is a stack.
- `queue` if it is certain that the box is a queue.
- `priority queue` if it is certain that the box is a max priority queue.
- `impossible` if it is certain that the box cannot be any of those three data structures.
- `not sure` if the box could possibly be more than one of those three data structures.

Example 1

```
Input:
6
1 1
1 2
1 3
2 1
2 2
2 3

Output:
queue
```

Example 2

```
Input:
6
1 1
1 2
1 3
2 3
2 2
2 1

Output:
not sure
```

Example 3

```
Input:
2
1 1
2 2

Output:
impossible
```

Example 4

```
Input:
7
1 84
1 36
1 61
1 4
2 4
1 61
2 61

Output:
stack
```

Example 5

```
Input:
7
1 2
1 5
1 1
1 3
2 5
1 4
2 4

Output:
priority queue
```