CODEFORCES β
Sponsored by Telegram

Codeforces Round #356 (Div. 2)

A. Bear and Five Cards

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A little bear Limak plays a game. He has five cards. There is one number written on each card. Each number is a positive integer.

Limak can discard (throw out) some cards. His goal is to minimize the sum of numbers written on remaining (not discarded) cards.

He is allowed to **at most once** discard two or three cards with the same number. Of course, he won't discard cards if it's impossible to choose two or three cards with the same number.

Given five numbers written on cards, cay you find the minimum sum of numbers on remaining cards?

Input
The only line of the input contains five integers t_1, t_2, t_3, t_4 and t_5 ($1 \le t_i \le 100$) — numbers written on cards.

Output
Print the minimum possible sum of numbers written on remaining cards.

Examples

input
7 3 7 3 20
output
26

input
7 9 3 1 8
output
28

input
10 10 10 10 10
output
20

Note
In the first sample, Limak has cards with numbers 7, 3, 7, 3 and 20. Limak can do one of the following.

- Do nothing and the sum would be $7 + 3 + 7 + 3 + 20 = 40$.
- Remove two cards with a number 7. The remaining sum would be $3 + 3 + 20 = 26$.
- Remove two cards with a number 3. The remaining sum would be $7 + 7 + 20 = 34$.

You are asked to minimize the sum so the answer is 26.

In the second sample, it's impossible to find two or three cards with the same number. Hence, Limak does nothing and the sum is $7 + 9 + 1 + 3 + 8 = 28$.

In the third sample, all cards have the same number. It's optimal to discard any three cards. The sum of two remaining numbers is $10 + 10 = 20$.

B. Bear and Finding Criminals

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n cities in Bearland, numbered 1 through n. Cities are arranged in one long row. The distance between cities i and j is equal to $|i - j|$.

Limak is a police officer. He lives in a city a. His job is to catch criminals. It's hard because he doesn't know in which cities criminals are. Though, he knows that there is **at most one** criminal in each city.

Limak is going to use a BCD (Bear Criminal Detector). The BCD will tell Limak how many criminals there are for every distance from a city a. After that, Limak can catch a criminal in each city for which he **is sure** that there must be a criminal.

You know in which cities criminals are. Count the number of criminals Limak will catch, after he uses the BCD.

Input

The first line of the input contains two integers n and a ($1 \le a \le n \le 100$) — the number of cities and the index of city where Limak lives.

The second line contains n integers $t_1, t_2, ..., t_n$ ($0 \le t_i \le 1$). There are t_i criminals in the i-th city.

Output

Print the number of criminals Limak will catch.

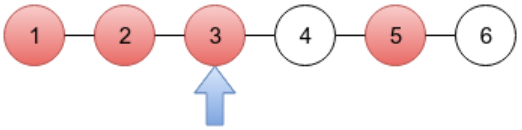
Examples

input
6 3 1 1 1 0 1 0
output
3

input
5 2 0 0 0 1 0
output
1

Note

In the first sample, there are six cities and Limak lives in the third one (blue arrow below). Criminals are in cities marked red.

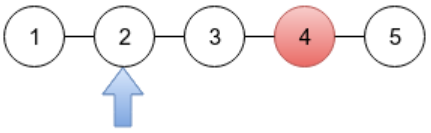


Using the BCD gives Limak the following information:

- There is one criminal at distance 0 from the third city — Limak is sure that this criminal is exactly in the third city.
- There is one criminal at distance 1 from the third city — Limak doesn't know if a criminal is in the second or fourth city.
- There are two criminals at distance 2 from the third city — Limak is sure that there is one criminal in the first city and one in the fifth city.
- There are zero criminals for every greater distance.

So, Limak will catch criminals in cities 1, 3 and 5, that is 3 criminals in total.

In the second sample (drawing below), the BCD gives Limak the information that there is one criminal at distance 2 from Limak's city. There is only one city at distance 2 so Limak is sure where a criminal is.



C. Bear and Prime 100

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem. In the output section below you will see the information about flushing the output.

Bear Limak thinks of some hidden number — an integer from interval $[2, 100]$. Your task is to say if the hidden number is prime or composite.

Integer $x > 1$ is called prime if it has exactly two distinct divisors, 1 and x. If integer $x > 1$ is not prime, it's called composite.

You can ask up to 20 queries about divisors of the hidden number. In each query you should print an integer from interval $[2, 100]$. The system will answer "`yes`" if your integer is a divisor of the hidden number. Otherwise, the answer will be "`no`".

For example, if the hidden number is 14 then the system will answer "`yes`" only if you print 2, 7 or 14.

When you are done asking queries, print "`prime`" or "`composite`" and terminate your program.

You will get the `Wrong Answer` verdict if you ask more than 20 queries, or if you print an integer not from the range $[2, 100]$. Also, you will get the `Wrong Answer` verdict if the printed answer isn't correct.

You will get the `Idleness Limit Exceeded` verdict if you don't print anything (but you should) or if you forget about flushing the output (more info below).

Input

After each query you should read one string from the input. It will be "`yes`" if the printed integer is a divisor of the hidden number, and "`no`" otherwise.

Output

Up to 20 times you can ask a query — print an integer from interval $[2, 100]$ in one line. You have to both print the end-of-line character and `flush` the output. After flushing you should read a response from the input.

In any moment you can print the answer "`prime`" or "`composite`" (without the quotes). After that, flush the output and terminate your program.

To flush you can use (just after printing an integer and end-of-line):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- See the documentation for other languages.

Hacking. To hack someone, as the input you should print the hidden number — one integer from the interval $[2, 100]$. Of course, his/her solution won't be able to read the hidden number from the input.

Examples

input
yes no yes

output
2 80 5 composite

input
no yes no no no

output
58 59 78 78 2 prime

Note

The hidden number in the first query is 30. In a table below you can see a better form of the provided example of the communication process.

solution	system
2	yes
80	no
5	yes
composite	

The hidden number is divisible by both 2 and 5. Thus, it must be composite. Note that it isn't necessary to know the exact value of the hidden number. In this test, the hidden number is 30.

solution	system
58	no
59	yes
78	no
78	no
2	no
prime	

59 is a divisor of the hidden number. In the interval $[2, 100]$ there is only one number with this divisor. The hidden number must be 59, which is prime. Note that the answer is known even after the second query and you could print it then and terminate. Though, it isn't forbidden to ask unnecessary queries (unless you exceed the limit of 20 queries).

D. Bear and Tower of Cubes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is a little polar bear. He plays by building towers from blocks. Every block is a cube with positive integer length of side. Limak has infinitely many blocks of each side length.

A block with side a has volume a^3. A tower consisting of blocks with sides $a_1, a_2, ..., a_k$ has the total volume $a_1^3 + a_2^3 + ... + a_k^3$.

Limak is going to build a tower. First, he asks you to tell him a positive integer X — the required total volume of the tower. Then, Limak adds new blocks greedily, one by one. Each time he adds the biggest block such that the total volume doesn't exceed X.

Limak asks you to choose X not greater than m. Also, he wants to maximize the number of blocks in the tower at the end (however, he still behaves greedily). Secondarily, he wants to maximize X.

Can you help Limak? Find the maximum number of blocks his tower can have and the maximum $X \le m$ that results this number of blocks.

Input

The only line of the input contains one integer m ($1 \le m \le 10^{15}$), meaning that Limak wants you to choose X between 1 and m, inclusive.

Output

Print two integers — the maximum number of blocks in the tower and the maximum required total volume X, resulting in the maximum number of blocks.

Examples

input
48
output
9 42

input
6
output
6 6

Note

In the first sample test, there will be 9 blocks if you choose $X = 23$ or $X = 42$. Limak wants to maximize X secondarily so you should choose 42.

In more detail, after choosing $X = 42$ the process of building a tower is:

- Limak takes a block with side 3 because it's the biggest block with volume not greater than 42. The remaining volume is $42 - 27 = 15$.
- The second added block has side 2, so the remaining volume is $15 - 8 = 7$.
- Finally, Limak adds 7 blocks with side 1, one by one.

So, there are 9 blocks in the tower. The total volume is $3^3 + 2^3 + 7 \cdot 1^3 = 27 + 8 + 7 = 42$.

E. Bear and Square Grid

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a grid with n rows and n columns. Each cell is either empty (denoted by '.') or blocked (denoted by 'X').

Two empty cells are *directly connected* if they share a side. Two cells (r_1, c_1) (located in the row r_1 and column c_1) and (r_2, c_2) are *connected* if there exists a sequence of empty cells that starts with (r_1, c_1), finishes with (r_2, c_2), and any two consecutive cells in this sequence are directly connected. A *connected component* is a set of empty cells such that any two cells in the component are connected, and there is no cell in this set that is connected to some cell not in this set.

Your friend Limak is a big grizzly bear. He is able to destroy any obstacles in some range. More precisely, you can choose a square of size $k \times k$ in the grid and Limak will transform all blocked cells there to empty ones. However, you can ask Limak to help only once.

The chosen square must be completely inside the grid. It's possible that Limak won't change anything because all cells are empty anyway.

You like big connected components. After Limak helps you, what is the maximum possible size of the biggest connected component in the grid?

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 500$) — the size of the grid and Limak's range, respectively.

Each of the next n lines contains a string with n characters, denoting the i-th row of the grid. Each character is '.' or 'X', denoting an empty cell or a blocked one, respectively.

Output

Print the maximum possible size (the number of cells) of the biggest connected component, after using Limak's help.

Examples

input
5 2 ..XXX XX.XX X.XXX X...X XXXX.
output
10

input
5 3 ..... .XXX. .XXX. .XXX. .....
output
25

Note

In the first sample, you can choose a square of size 2×2. It's optimal to choose a square in the red frame on the left drawing below. Then, you will get a connected component with 10 cells, marked blue in the right drawing.