Constellation

Problem Description

Three characters { #, *, . } represents a constellation of stars and galaxies in space. Each galaxy is demarcated by # characters. There can be one or many stars in a given galaxy. Stars can only be in shape of vowels { A, E, I, O, U } . A collection of * in the shape of the vowels is a star. A star is contained in a 3x3 block. Stars cannot be overlapping. The dot(.) character denotes empty space.

Given 3xN matrix comprising of { #, *, . } character, find the galaxy and stars within them.

Note: Please pay attention to how vowel ***A*** is denoted in a 3x3 block in the examples section below.
Constraints

3 <= N <= 10^5

Input

Input consists of single integer N denoting number of columns.

Output

Output contains vowels (stars) in order of their occurrence within the given galaxy. Galaxy itself is represented by # character.

Time Limit

1

Examples

Example 1

Input

18

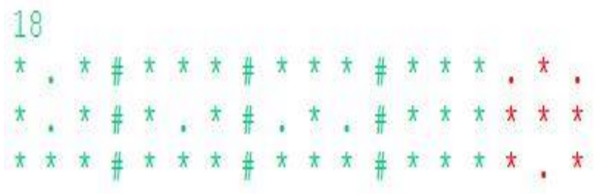
```
* . * # * * * # * * * # * * * . * .

* . * # * . * # . * . # * * * * * *

* * * # * * * # * * * # * * * * . *
```

Output

U#O#I#EA

Explanation

As it can be seen that the stars make the image of the alphabets U, O, I, E and A respectively.


```
18
* . * # * * * # * * * # * * * . * .
* . * # * . * # . * . # * * * * * *
* * * # * * * # * * * # * * * * . *
```

Example 2

Input

12

* . * # . * * * # . * .

* . * # . . * . # * * *

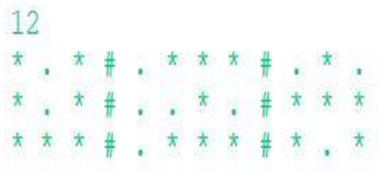
* * * # . * * * # * . *

Output

U#I#A

Explanation

As it can be seen that the stars make the image of the alphabet U, I and A.


```
12
* . * # . * * * # . * .
* . * # . . * . # * * *
* * * # . * * * # * . *
```

Prime Time Again

Problem Description

Here on earth, our 24-hour day is composed of two parts, each of 12 hours. Each hour in each part has a corresponding hour in the other part separated by 12 hours: the hour essentially measures the duration since the start of the day part. For example, 1 hour in the first part of the day is equivalent to 13, which is 1 hour into the second part of the day.

Now, consider the equivalent hours that are both prime numbers. We have 3 such instances for a 24-hour 2-part day:

5~17

7~19

11~23

Accept two natural numbers D, P >1 corresponding respectively to number of hours per day and number of parts in a day separated by a space. D should be divisible by P, meaning that the number of hours per part (D/P) should be a natural number. Calculate the number of instances of equivalent prime hours. Output zero if there is no such instance. Note that we require each equivalent hour in each part in a day to be a prime number.

Example:

Input: 24 2

Output: 3 (We have 3 instances of equivalent prime hours: 5~17, 7~19 and 11~23.)

Constraints

10 <= D < 500

2 <= P < 50

Input

Single line consists of two space separated integers, D and P corresponding to number of hours per day and number of parts in a day respectively

Output
Output must be a single number, corresponding to the number of instances of equivalent prime number, as described above
Time Limit

1

Examples

Example 1

Input

36 3

Output

2

Explanation

In the given test case D = 36 and P = 3

Duration of each day part = 12

2~14~X

3~15~X

5~17~29 - instance of equivalent prime hours

7~19~31 - instance of equivalent prime hours

11~23~X

Hence the answers is 2.

Example 2

Input

49 7

Output

0

Explanation

Duration of each day part = 7

2~9~X~23~X~37~X

3~X~17~X~31~X~X

5~X~19~X~X~X~47

7~X~X~X~X~X~X

Hence there are no equivalent prime hours.

Minimize The Sum

Problem Description

Given an array of integers, perform atmost K operations so that the sum of elements of final array is minimum. An operation is defined as follows -

Consider any 1 element from the array, arr[i].

Replace arr[i] by floor(arr[i]/2).

Perform next operations on updated array.

The task is to minimize the sum after atmost K operations.

Constraints

1 <= N, K <= 10^5.

Input

First line contains two integers N and K representing size of array and maximum numbers of operations that can be performed on the array respectively.

Second line contains N space separated integers denoting the elements of the array, arr.

Output

Print a single integer denoting the minimum sum of the final array.

Time Limit

1

Examples

Example 1

Input

4 3

20 7 5 4

Output

17

Explanation

Operation 1 -> Select 20. Replace it by 10.

New array = [10, 7, 5, 4]

Operation 2 -> Select 10. Replace it by 5.

New array = [5, 7, 5, 4].

Operation 3 -> Select 7. Replace it by 3.

New array = [5,3,5,4].

Sum = 17.

Critical Planets

Problem Description

War between Republic and Separatist is escalating. The Separatist are on a new offensive. They have started blocking the path between the republic planets (represented by integers), so that these planets surrender due to the shortage of food and supplies. The Jedi council has taken a note of the situation and they have assigned Jedi Knight Skywalker and his Padawan Ahsoka to save the critical planets from blockade (Those planets or system of planets which can be accessed by only one path and may be lost if that path is blocked by separatist).

Skywalker is preparing with the clone army to defend the critical paths. He has assigned Ahsoka to find the critical planets. Help Ahsoka to find the critical planets(C) in ascending order. You only need to specify those planets which have only one path between them and they cannot be accessed by any other alternative path if the only path is compromised.

Constraints

M <= 10000

N <= 7000

Input

First line contains two space separated integers M and N, where M denotes the number of paths between planets and N denotes the number of planets.

Next M lines, each contains two space separated integers, representing the planet numbers that have a path between them.

Output

C lines containing one integer representing the critical planet that they need to save in ascending order of the planet number if no planet is critical then print -1

Time Limit

1

Examples

Example 1

Input

3 4

0 1

1 2

2 3

Output

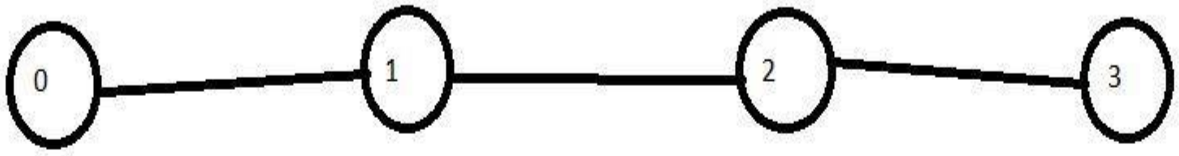
0

1

2

3

Explanation



Since all the planets are connected with one path and cannot be accessed by any alternative paths hence all the planets are critical.

Example 2

Input

7 6

0 2

0 1

1 2

2 3

4 5

3 4

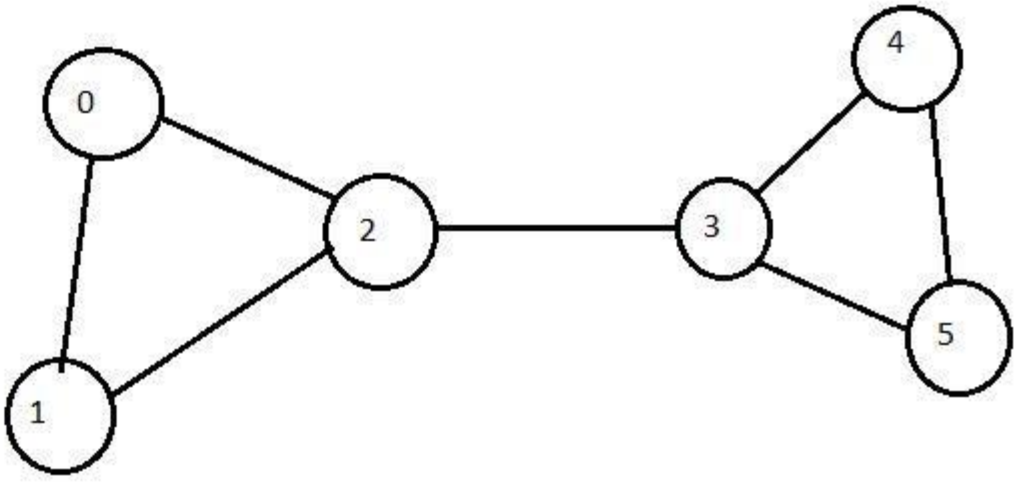
3 5

Output

2

3

Explanation



If the republic loose the path between 2 and 3 then the two system of planets will not be able to communicate with each other. Hence 2 and 3 are critical planets.

Paste Reduction

Problem Description

Some keys in Codu's computer keyboard are not working. Fortunately for Codu, these characters are present in a previously existing text file.

He wants to write a paragraph which involves typing those characters whose keys are defunct in Codu's keyboard. Only option left for Codu is to copy-paste those characters from the previously existing text files. However, copy-pasting keys is a laborious operation since one has to switch windows and also previously copied items are lost once a new set of characters are copied. Hence, Codu wants to minimize the number of times he needs to copy-paste from that text file. Fortunately there can be situations where previously copied characters are readily available for pasting. Help Codu devise a method to minimize the number of times a paste operation is needed, given - the text he intends to type and the faulty keys

Constraints

0 < Length of paragraph <= 1000 characters

0 < number of faulty keys in keyboard <= 36

Only a to z and 0 to 9 keys can be faulty.

Input paragraph will not contain any upper-case letter.

Input

First line contains a paragraph that is be to be written.

Second line contains a string. This string has to be interpreted in the following fashion

All characters in that string correspond to faulty keys

That string is available for copy as-is from the other text file that Codu is referring

Also, individual characters can always be copied from the same text file

Output

Single integer denoting minimum number of copy operations required

Time Limit

1

Examples

Example 1

Input

supreme court is the highest judicial court

su

Output

4

Explanation

Codu will first paste su from the file when typing characters su in the word supreme.

In the second instance, Codu will need to copy character u and paste it when typing character u in the word court.

In the third instance, Codu will need to copy character su and paste su when typing character s in the word is. Codu will back track using the left arrow key and type the characters "the highe".

In the fourth instance, Codu will again paste su. The overall string typed until this moment is "supreme court is the highesuu". Codu will then back track again using left arrow key and type characters "t j". At this point the typed string is "supreme court is the highest juu". Cursor is after character j. Codu will use right arrow key and now the cursor will be after ju. Codu will now type "udicial co". String typed till this point is "supreme court is the highest judicial cou". Cursor is after character o. Codu will use right arrow and the cursor will be after character u. Finally Codu will type "rt".

Final string - "supreme court is the highest judicial court" - is thus fully typed. Here, the number of paste operations are 4.

Maximum Prize

Problem Description

Imagine you are a martial arts fighter fighting with fellow martial artists to win prize money. However unlike traditional competitions, here you have the opportunity to pick and choose your opponent to maximize your prize kitty. The rules of maximization of prize kitty are as follows

► You have a superpower bestowed upon you, that you will win against anyone you challenge

► You have to choose the right order because unfortunately the superpower does not ensure that your prize money is always the highest

► Every victory against an opponent that you challenge and win against, will translate into a certain winning sum

► Here begins the technical part that you need to know in order to maximize your winning prize money

- All your opponents are standing in one line next to each other i.e. the order of opponents is fixed
- Your first task is to choose a suitable opponent from this line
- When you choose one opponent from that line, he steps out of the line and fights you.
- After you beat him, you get to decide how your prize money for winning against him will be calculated
- Essentially, if the opponent you have beaten has two neighbours, then you have the option to multiply the *opponent number* with any one of the two neighbours and add the other *opponent number.* That value becomes your prize money for that match
- If your opponent has only one neighbor then your prize money for that match is product of current *opponent number* with neighbours' *opponent number*
- When dealing with last opponent in the tournament, your prize money is equal to the value of the last *opponent number*
- As the tournament proceeds, the opponent that you have beaten has to leave the tournament

Example: 2 5 6 7

This depicts that you have four opponents with numbers 2 5 6 and 7 respectively

1. Suppose you choose to fight opponent number 5, then after winning, the max prize kitty you can win for that match is = 5*6+2 = 32

   Now opponent number 5 is out of the game. So opponent number 2 6 7 remain

2. Suppose you now choose to fight opponent number 2, then after winning, the max prize kitty you can win for that match is = 2*6+0 = 12. Your overall prize kitty is now 32 + 12 = 44

   Now opponent number 2 is out of the game. So opponent number 6 7 remain

3. Suppose you now choose to fight opponent number 6, then after winning, the max prize kitty you can win for that match is = 7*6+0 = 42. Your overall prize kitty is now 44 + 42 = 86

Now opponent number 6 is out of the game. So opponent number 7 remains

4. After beating opponent number 7, the max prize kitty you can win for that match is 7

So overall prize kitty in this case is 93.

Other orders of choosing opponents will yield the following overall prize kitty

- Order 7->2->6->5 will yield overall prize kitty as 87
- Order 2->5->6->7 will yield overall prize kitty as 88
- Order 5->6->2->7 will yield overall prize kitty as 95
- Order 6- >7->2->5 will yield overall prize kitty as 97
- But by following the order 6->5->2->7 will yield overall prize kitty as 105, which is maximum.

Your task is to maximize your prize kitty by taking the right decisions

Input

First line contains an integer N which denotes the number of opponents in the tournament

Second line contains N space separated integers, which are the *opponent numbers* of other opponents
Output

Print the maximum number of coins you can win

Constraints

1 <= N <= 500

0 <= individual coin count < 100
Time Limit

1

Examples

Example 1

Input

4

2 5 6 7

Output

105

Explanation:

Refer the explanation in problem description.

Example 2

Input

3

7 8 9

Output

151

Explanation:

1. You choose to fight opponent number 8, then after winning, the max prize kitty you can win for that match is = 8*9+7 = 79

Now opponent number 8 is out of the game. So opponent number 7 9 remain

2. Suppose you now choose to fight opponent number 7, then after winning, the max prize kitty you can win for that match is = 7*9+0 = 63. Your overall prize kitty is now 79 + 63 = 142

Now opponent number 7 is out of the game. So opponent number 9 remains

3. After beating opponent number 9, the max prize kitty you can win for that match is 9

So overall prize kitty in this case is 142 + 9 = 151.