# **PROBLEM SOLVING DATABASE**



**ID: 25000000**

1. **Data Structures [25100000]**
   1. ***Arrays [25110000]***
      1. [25110001] There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

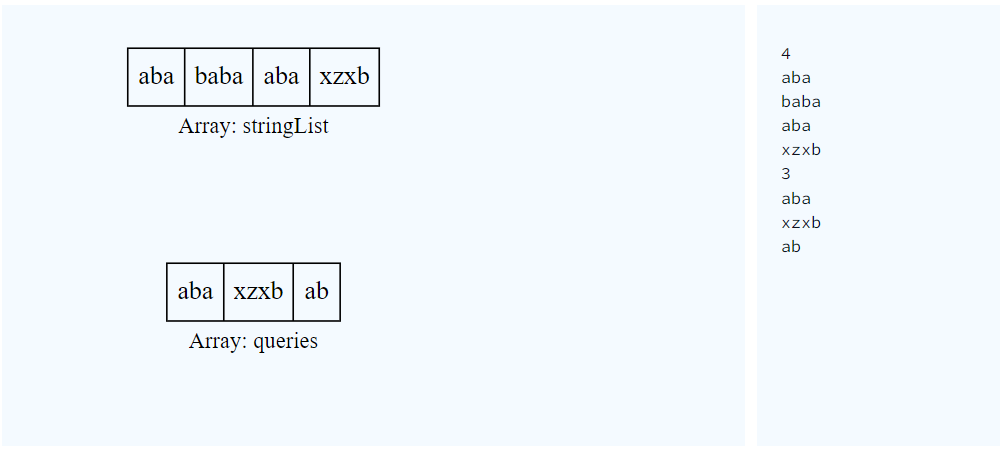
**Example**



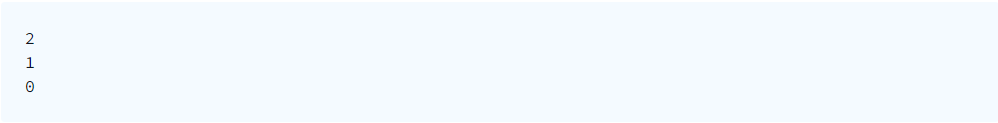
There are 2 instances of *‘ab’*, 1 of *‘abc’*, and 0 of *‘bc’* for each query, add an element to the return array,



**Sample Input 1**



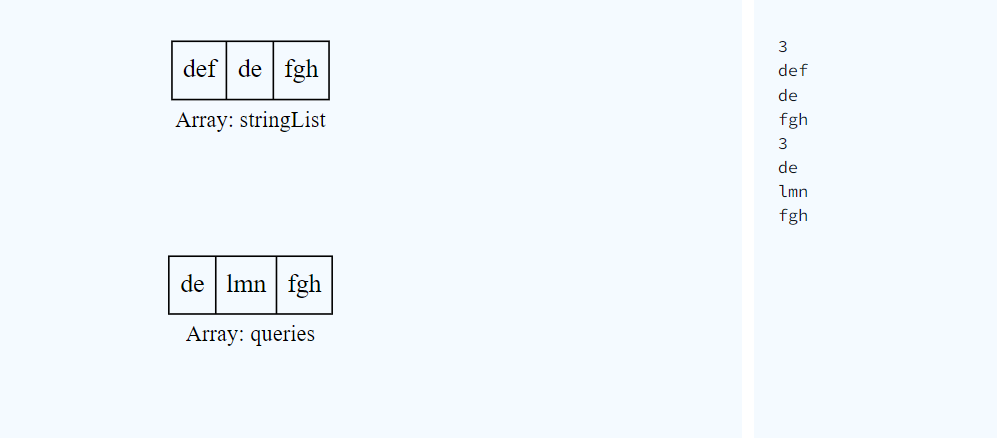
**Sample Output 1**



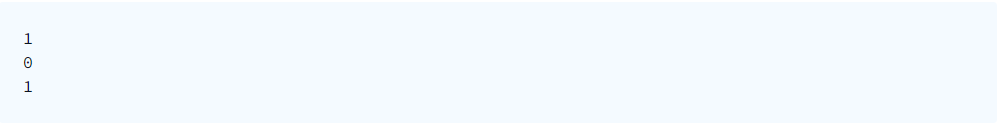
**Explanation 1**

Here, "aba" occurs twice, in the first and third string. The string "xzxb" occurs once in the fourth string, and "ab" does not occur at all.

**Sample Input 2**



**Sample Output 2**



**Sample Input 3**

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

**Sample Output 3**

Ảnh có chứa ảnh chụp màn hình, hàng

Mô tả được tạo tự động

* + 1. [25110002] Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in the array.

**Example**



Queries are interpreted as follows:

Ảnh có chứa ảnh chụp màn hình, màu trắng, thiết kế

Mô tả được tạo tự động

Add the values of k between the indices a and b inclusive:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, đại số

Mô tả được tạo tự động

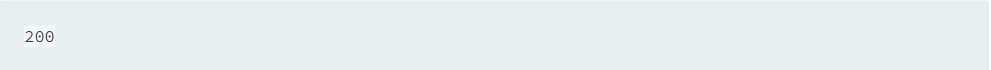
The largest value is 10 after all operations are performed.

**Sample Input**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu trắng

Mô tả được tạo tự động

**Sample Output**



**Explanation**

After the first update the list is 100 100 0 0 0.

After the second update list is 100 200 100 100 100.

After the third update list is 100 200 200 200 100.

The maximum value is 200.

* + 1. [25110003] A left rotation operation on an array of size n shifts each of the array's elements 1 unit to the left. Given an integer, d, rotate the array that many steps left and return the result.

**Example**



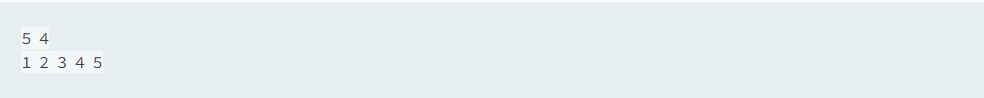
After 2 rotations,

****

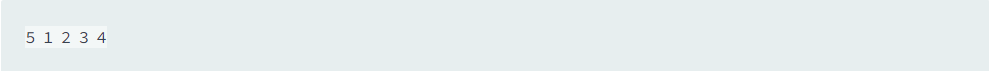
**Constraints**



**Sample Input**



**Sample Output**



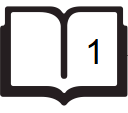
**Explanation**

To perform d = 4 left rotations, the array undergoes the following sequence of changes:



1. **Algorithms [25200000]**
   1. ***Implementation [25210000]***
      1. [25210001] **Drawing Books**

A teacher asks the class to open their books to a page number. A student can either start turning pages from the front of the book or from the back of the book. They always turn pages one at a time. When they open the book, page 1 is always on the right side:



When they flip page 1, they see pages 2 and 3. Each page except the last page will always be printed on both sides. The last page may only be printed on the front, given the length of the book. If the book is n pages long, and a student wants to turn to page p, what is the minimum number of pages to turn? They can start at the beginning or the end of the book.

Given n and p, find and print the minimum number of pages that must be turned in order to arrive at page p.

**Example**

  
Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, màu trắng

Mô tả được tạo tự động

Using the diagram above, if the student wants to get to page 3, they open the book to page 1, flip 1 page and they are on the correct page. If they open the book to the last page, page 5, they turn 1 page and are at the correct page. Return 1.

**Constraints**

****

**Sample Input 0**

6

2

**Sample Output 0**

1

**Explanation 0**

If the student starts turning from page 1, they only need to turn 1 page:

Ảnh có chứa biểu tượng, hàng, màu trắng, biểu đồ

Mô tả được tạo tự động

If a student starts turning from page 6, they need to turn 2 pages:

Ảnh có chứa Phông chữ, biểu tượng, màu trắng, Đồ họa

Mô tả được tạo tự động

Return the minimum value, 1.

**Sample Input 1**

5

4

**Sample Output 1**

0

**Explanation 1**

If the student starts turning from page 1, they need to turn 2 pages:

Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, màu trắng

Mô tả được tạo tự động

If they start turning from page 5, they do not need to turn any pages:

Ảnh có chứa biểu tượng, Phông chữ, thiết kế

Mô tả được tạo tự động

Return the minimum value, 0.

* + 1. [25210002] **Electronics Shop**

A person wants to determine the most expensive computer keyboard and USB drive that can be purchased with a give budget. Given price lists for keyboards and USB drives and a budget, find the cost to buy them. If it is not possible to buy *both* items, return -1.

**Example**  


The person can buy a **40 keyboard + 12 USB drive = 52**, or a  **50 keyboard + 8 USB drive = 58**. Choose the latter as the more expensive option and return 58.

**Input Format**

The first line contains three space-separated integers b, n, and m, the budget, the number of keyboard models and the number of USB drive models.

The second line contains n space-separated integers keyboard[i], the prices of each keyboard model.

The third line contains m space-separated integers drives, the prices of the USB drives.

**Constraints**

* The price of each item is in the inclusive range .

**Sample Input 0**

10 2 3

3 1

5 2 8

**Sample Output 0**

9

**Explanation 0**

Buy the  keyboard and the  USB drive for a total cost of 8 + 1 = 9.

**Sample Input 1**

5 1 1

4

5

**Sample Output 1**

-1

**Explanation 1**

There is no way to buy one keyboard and one USB drive because 4 + 5 > 5, so return -1.

* + 1. [25210003] **Cats and a Mouse**

Two cats and a mouse are at various positions on a line. You will be given their starting positions. Your task is to determine which cat will reach the mouse first, assuming the mouse does not move and the cats travel at equal speed. If the cats arrive at the same time, the mouse will be allowed to move and it will escape while they fight.

You are given q queries in the form of x, y, and z representing the respective positions for cats A and B, and for mouse C. Return the appropriate answer to each query, which will be printed on a new line.

* If cat A catches the mouse first, print Cat A.
* If cat B catches the mouse first, print Cat B.
* If both cats reach the mouse at the same time, print Mouse C as the two cats fight and mouse escapes.

**Example**

x = 2

y = 5

z = 4

The cats are at positions 2 (Cat A) and 5 (Cat B), and the mouse is at position 4. Cat B, at position 5 will arrive first since it is only 1 unit away while the other is 2 units away. Return 'Cat B'.

**Input Format**

The first line contains a single integer, q, denoting the number of queries.

Each of the q subsequent lines contains three space-separated integers describing the respective values of  x (cat A's location),  (cat B's location), and  (mouse C's location).

**Constraints**

**Sample Input 0**

2

1 2 3

1 3 2

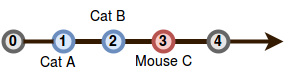
**Sample Output 0**

Cat B

Mouse C

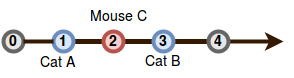
**Explanation 0**

*Query 0:* The positions of the cats and mouse are shown below:



Cat  will catch the mouse first, so we print Cat B on a new line.

*Query 1*: In this query, cats  and  reach mouse  at the exact same time:



Because the mouse escapes, we print Mouse C on a new line.

* + 1. [25210004] **Forming a Magic Square**

We define a [magic square](https://en.wikipedia.org/wiki/Magic_square) to be an n × n matrix of distinct positive integers from 1 to  where the sum of any row, column, or diagonal of length n is always equal to the same number: the *magic constant*.

You will be given a  matrix 3 × 3 of integers in the inclusive range [1, 9]. We can convert any digit a to any other digit b in the range [1, 9] at cost of |a – b|. Given s, convert it into a magic square at *minimal* cost. Print this cost on a new line.

**Note:** The resulting magic square must contain distinct integers in the inclusive range [1, 9].

**Example**

s = [[5, 3, 4], [1, 5, 8], [6, 4, 2]]

The matrix looks like this:

5 3 4

1 5 8

6 4 2

We can convert it to the following magic square:

8 3 4

1 5 9

6 7 2

This took three replacements at a cost of |5 – 8| + |8 – 9| + |4 – 7| = 7.

**Input Format**

Each of the 3 lines contains three space-separated integers of row s[i].

**Constraints**

* s[i][j] ∈ [1, 9]

**Sample Input 0**

4 9 2

3 5 7

8 1 5

**Sample Output 0**

1

**Explanation 0**

If we change the bottom right value, s[2][2], from 5 to 6 at a cost of |6 – 5| = 1, s becomes a magic square at the minimum possible cost.

**Sample Input 1**

4 8 2

4 5 7

6 1 6

**Sample Output 1**

4

**Explanation 1**

Using 0-based indexing, if we make

* s[0][1] -> 9 at a cost of |9 – 8| = 1
* s[1][0] -> 3 at a cost of |3 – 4| = 1
* s[2][0] -> 8 at a cost of |8 – 6| = 2,

then the total cost will be 1 +1 + 2 = 4.

* + 1. [25210005] **Picking Numbers**

Given an array of integers, find the longest subarray where the absolute difference between any two elements is less than or equal to 1.

**Example**

There are two subarrays meeting the criterion: [1, 1, 2, 2] and [4, 4, 5, 5, 5]. The maximum length subarray has 5 elements.

**Input Format**

The first line contains a single integer n, the size of the array .

The second line contains n space-separated integers, each an a[i].

**Constraints**

* The answer will be ≥ 2.

**Sample Input 0**

6

4 6 5 3 3 1

**Sample Output 0**

3

**Explanation 0**

We choose the following multiset of integers from the array: {4, 3, 3}. Each pair in the multiset has an absolute difference ≤ 1 (i.e., |4 – 3| = 1 and |3 – 3| = 0), so we print the number of chosen integers, 3, as our answer.

**Sample Input 1**

6

1 2 2 3 1 2

**Sample Output 1**

5

**Explanation 1**

We choose the following multiset of integers from the array: {1, 2, 2, 1, 2}. Each pair in the multiset has an absolute difference ≤ 1 (i.e., |1 – 2| = 0, |1 – 1| = 0, and |2 – 2| = 0), so we print the number of chosen integers, 5, as our answer.

* + 1. [25210006] **Climbing the Leaderboard**

An arcade game player wants to climb to the top of the leaderboard and track their ranking. The game uses [Dense Ranking](https://en.wikipedia.org/wiki/Ranking#Dense_ranking_.28.221223.22_ranking.29), so its leaderboard works like this:

* The player with the highest score is ranked number 1 on the leaderboard.
* Players who have equal scores receive the same ranking number, and the next player(s) receive the immediately following ranking number.

**Example**

ranked = [100, 90, 90, 80]

player = [70, 80, 105]

The ranked players will have ranks 1, 2, 2, and 3, respectively. If the player's scores are 70, 80 and 105, their rankings after each game are 4th , 3rd  and 1st. Return [4, 3, 1].

**Input Format**

The first line contains an integer n, the number of players on the leaderboard.

The next line contains n space-separated integers ranked[i], the leaderboard scores in decreasing order.

The next line contains an integer, m, the number games the player plays.

The last line contains m space-separated integers player[j], the game scores.

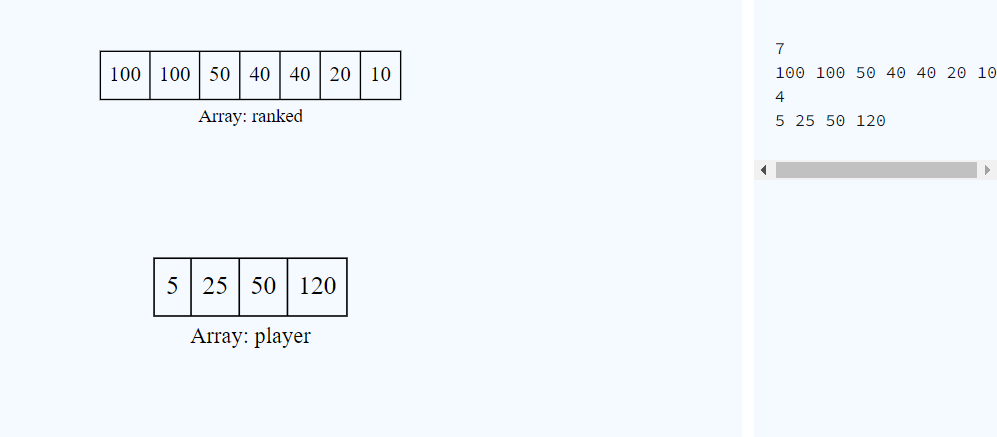
**Constraints**

* for
* for
* The existing leaderboard, ranked, is in *descending* order.
* The player's scores, player, are in *ascending* order.

**Subtask**

For  of the maximum score:

**Sample Input 1**

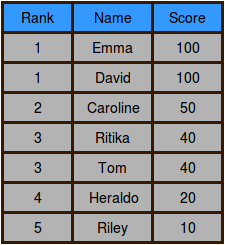
****

**Sample Output 1**

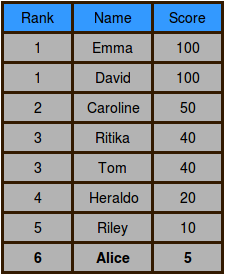
6  
4  
2  
1

**Explanation 1**

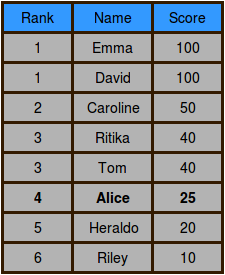
Alice starts playing with 7 players already on the leaderboard, which looks like this:



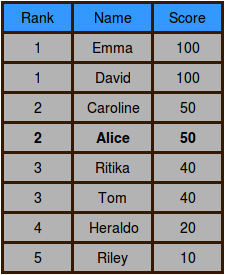
After Alice finishes game 0, her score is 5 and her ranking is 6:



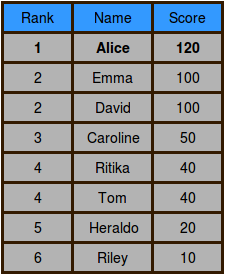
After Alice finishes game 1, her score is 25 and her ranking is 4:



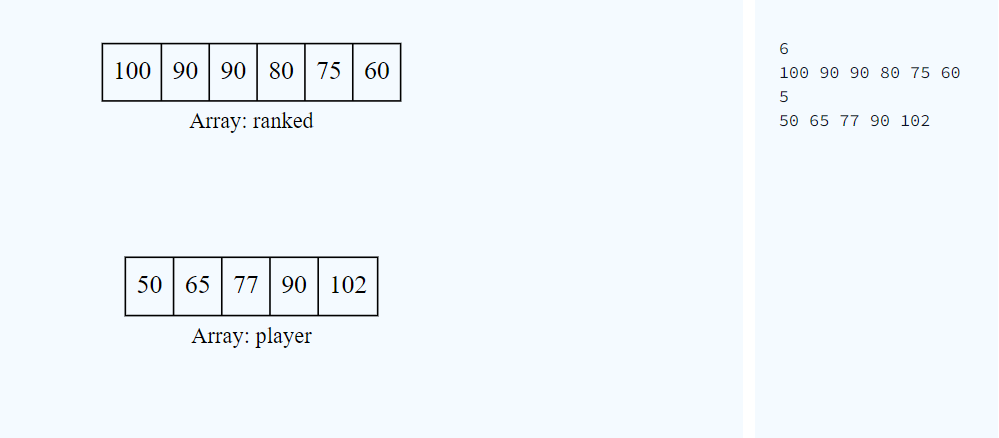
After Alice finishes game 2, her score is 50 and her ranking is tied with Caroline at 2:



After Alice finishes game 3, her score is 120 and her ranking is 1:



**Sample Input 2**

 **Sample Output 2**

6  
5  
4  
2  
1

* + 1. [25210007] **Save the Prisoner!**

A jail has a number of prisoners and a number of treats to pass out to them. Their jailer decides the fairest way to divide the treats is to seat the prisoners around a circular table in sequentially numbered chairs. A chair number will be drawn from a hat. Beginning with the prisoner in that chair, one candy will be handed to each prisoner sequentially around the table until all have been distributed.

The jailer is playing a little joke, though. The last piece of candy looks like all the others, but it tastes *awful*. Determine the chair number occupied by the prisoner who will receive that candy.

**Example**

n = 4

m = 6

s = 2

There are 4 prisoners,  6 pieces of candy and distribution starts at chair 2. The prisoners arrange themselves in seats numbered 1 to 4. Prisoners receive candy at positions 2, 3, 4, 1, 2, 3. The prisoner to be warned sits in chair number 3.

**Constraints**

**Sample Input 0**

2

5 2 1

5 2 2

**Sample Output 0**

2

3

**Explanation 0**

In the first query, there are n = 5 prisoners and m = 2 sweets. Distribution starts at seat number s = 1. Prisoners in seats numbered 1 and 2 get sweets. Warn prisoner 2.

In the second query, distribution starts at seat 2 so prisoners in seats 2 and 3 get sweets. Warn prisoner 3.

**Sample Input 1**

2

7 19 2

3 7 3

**Sample Output 1**

6

3

**Explanation 1**

In the first test case, there are n = 7 prisoners, m = 10 sweets and they are passed out starting at chair s = 2. The candies go all around twice and there are 5 more candies passed to each prisoner from seat 2 to seat 6.

In the second test case, there are n = 3 prisoners, m = 7 candies and they are passed out starting at seat s = 3. They go around twice, and there is one more to pass out to the prisoner at seat 3.

* + 1. [25210008] **Circular Array Rotation**

John Watson knows of an operation called a *right circular rotation* on an array of integers. One rotation operation moves the last array element to the first position and shifts all remaining elements right one. To test Sherlock's abilities, Watson provides Sherlock with an array of integers. Sherlock is to perform the rotation operation a number of times then determine the value of the element at a given position.

For each array, perform a number of right circular rotations and return the values of the elements at the given indices.

**Example**  
a = [3, 4, 5]

k = 2

queries = [1, 2]

Here k is the number of rotations on a, and queries holds the list of indices to report. First we perform the two rotations: [3, 4, 5] → [5, 3, 4] → [4, 5, 3]

Now return the values from the zero-based indices 1 and 2 as indicated in the queries array.

**Input Format**

The first line contains  space-separated integers, n, k, and q, the number of elements in the integer array, the rotation count and the number of queries.

The second line contains n space-separated integers, where each integer i describes array element a[i] (where 0 ≤ i ≤ n).  
Each of the q subsequent lines contains a single integer, queries[i], an index of an element in a to return.

**Constraints**

**Sample Input 0**

3 2 3

1 2 3

0

1

2

**Sample Output 0**

2

3

1

**Explanation 0**

After the first rotation, the array is [3, 1, 2].

After the second (and final) rotation, the array is [2, 3, 1].

We will call this final state array b = [2, 3, 1]. For each query, we just have to get the value of .

1. , .
2. , .
3. , .
   * 1. [25210009] **Sequence Equation**

Given a sequence of n integers,  where each element is distinct and satisfies . For *each* x where , that is x increments from 1 to n, find any integer y such that  and keep a history of the values of y in a return array.

**Example**

P = [5, 2, 1, 3, 4]

Each value of x between 1 and 5, the length of the sequence, is analyzed as follows:

1. , so
2. , so
3. , so
4. , so
5. , so

The values for y are [4, 2, 5, 1, 3].

**Input Format**

The first line contains an integer n, the number of elements in the sequence.

The second line contains n space-separated integers p[i] where 1 ≤ i ≤ n.

**Constraints**

* , where .
* Each element in the sequence is distinct.

**Sample Input 0**

3

2 3 1

**Sample Output 0**

2

3

1

**Explanation 0**

Given the values of , , and , we calculate and print the following values for each x from 1 to n:

1. , so we print the value of y = 2 on a new line.
2. , so we print the value of y = 3 on a new line.
3. , so we print the value of y = 1 on a new line.

**Sample Input 1**

5

4 3 5 1 2

**Sample Output 1**

1

3

5

4

2

* + 1. [25210010] **Extra Long Factorials**

The *factorial* of the integer , written , is defined as:

Calculate and print the factorial of a given integer.

For example, if n = 30, we calculate  and get 265252859812191058636308480000000.

**Note:** Factorials of  can't be stored even in a  long long variable. Big integers must be used for such calculations. Languages like Java, Python, Ruby etc. can handle big integers, but we need to write additional code in C/C++ to handle huge values.

We recommend solving this challenge using BigIntegers.

**Input Format**

Input consists of a single integer n

**Constraints**

**Output Format**

Print the factorial of n.

**Sample Input**

25

**Sample Output**

15511210043330985984000000

**Explanation**

* + 1. [25210011] **Append and Delete**

You have two strings of lowercase English letters. You can perform two types of operations on the first string:

1. *Append* a lowercase English letter to the end of the string.
2. *Delete* the last character of the string. Performing this operation on an empty string results in an empty string.

Given an integer, k, and two strings, s and t, determine whether or not you can convert s to t by performing *exactly* k of the above operations on s. If it's possible, print Yes. Otherwise, print No.

**Example**.   
s = [a, b, c]

t = [d, e, f]

k = 6

To convert s to t, we first delete all of the characters in 3 moves. Next we add each of the characters of t in order. On the 6th move, you will have the matching string. Return Yes.

If there were more moves available, they could have been eliminated by performing multiple deletions on an empty string. If there were fewer than 6 moves, we would not have succeeded in creating the new string.

**Input Format**

The first line contains a string s, the initial string.

The second line contains a string t, the desired final string.

The third line contains an integer k, the number of operations.

**Constraints**

* s and t consist of lowercase English letters, ascii[a-z].

**Sample Input 0**

hackerhappy

hackerrank

9

**Sample Output 0**

Yes

**Explanation 0**

We perform 5 delete operations to reduce string s to hacker. Next, we perform 4 append operations (i.e., r, a, n, and k), to get hackerrank. Because we were able to convert s to t by performing exactly k = 9 operations, we return Yes.

**Sample Input 1**

aba

aba

7

**Sample Output 1**

Yes

**Explanation 1**

We perform 4 delete operations to reduce string s to the empty string. Recall that though the string will be empty after 3 deletions, we can still perform a delete operation on an empty string to get the empty string. Next, we perform 3 append operations (i.e., a, b, and a). Because we were able to convert s to t by performing exactly k = 7 operations, we return Yes.

**Sample Input 2**

ashley

ash

2

**Sample Output 2**

No

**Explanation 2**

To convert ashley to ash a minimum of 3 steps are needed. Hence we print No as answer.

* + 1. [25210012] **Sherlock and Squares**

Watson likes to challenge Sherlock's math ability. He will provide a starting and ending value that describe a range of integers, inclusive of the endpoints. Sherlock must determine the number of *square integers* within that range.

**Note**: A square integer is an integer which is the square of an integer, e.g. 1, 4, 9, 16, 25.

**Example**  
a = 24

b = 49

There are three square integers in the range: 25, 36 and 49. Return 3.

**Input Format**

The first line contains , the number of test cases.

Each of the next  lines contains two space-separated integers,  and , the starting and ending integers in the ranges.

**Constraints**

**Sample Input**

2

3 9

17 24

**Sample Output**

2

0

**Explanation**

*Test Case #00:* In range [3, 9], 4 and 9 are the two square integers.

*Test Case #01:* In range [17, 34], there are no square integers.

* + 1. [25210013] **Equalize the Array**

Given an array of integers, determine the minimum number of elements to delete to leave only elements of equal value.

**Example**

Delete the 2 elements 1 and 3 leaving arr = [2, 2]. If both twos plus either the 1 or the 3 are deleted, it takes 3 deletions to leave either [3] or [1]. The minimum number of deletions is 2.

**Input Format**

The first line contains an integer n, the number of elements in arr.

The next line contains n space-separated integers arr[i].

**Constraints**

**Sample Input**

STDIN Function

----- --------

5 arr[] size n = 5

3 3 2 1 3 arr = [3, 3, 2, 1, 3]

**Sample Output**

2

**Explanation**

Delete arr[2] = 2 and arr[3] = 1 to leave arr’ = [3, 3, 3]. This is minimal. The only other options are to delete 4 elements to get an array of either [1] or [2].

* + 1. [25210014] **Cut the Sticks**

You are given a number of sticks of varying lengths. You will iteratively cut the sticks into smaller sticks, discarding the shortest pieces until there are none left. At each iteration you will determine the length of the shortest stick remaining, cut that length from each of the longer sticks and then discard all the pieces of that shortest length. When all the remaining sticks are the same length, they cannot be shortened so discard them.

Given the lengths of n sticks, print the number of sticks that are left before each iteration until there are none left.

**Example**  
arr = [1, 2, 3]

The shortest stick length is 1, so cut that length from the longer two and discard the pieces of length 1. Now the lengths are arr = [1, 2]. Again, the shortest stick is of length 1, so cut that amount from the longer stick and discard those pieces. There is only one stick left, arr = [1], so discard that stick. The number of sticks at each iteration are answer = [3, 2, 1].

**Input Format**

The first line contains a single integer n, the size of arr.

The next line contains n space-separated integers, each an arr[i], where each value represents the length of the iih stick.

**Constraints**

**Sample Input 0**

STDIN Function

----- --------

6 arr[] size n = 6

5 4 4 2 2 8 arr = [5, 4, 4, 2, 2, 8]

**Sample Output 0**

6

4

2

1

**Explanation 0**

sticks-length length-of-cut sticks-cut

5 4 4 2 2 8 2 6

3 2 2 \_ \_ 6 2 4

1 \_ \_ \_ \_ 4 1 2

\_ \_ \_ \_ \_ 3 3 1

\_ \_ \_ \_ \_ \_ DONE DONE

**Sample Input 1**

8

1 2 3 4 3 3 2 1

**Sample Output 1**

8

6

4

1

**Explanation 1**

sticks-length length-of-cut sticks-cut

1 2 3 4 3 3 2 1 1 8

\_ 1 2 3 2 2 1 \_ 1 6

\_ \_ 1 2 1 1 \_ \_ 1 4

\_ \_ \_ 1 \_ \_ \_ \_ 1 1

\_ \_ \_ \_ \_ \_ \_ \_ DONE DONE

* + 1. [25210015] **Non – Divisible Subset**

Given a set of distinct integers, print the size of a maximal subset of S where the sum of any 2 numbers in S’ is *not* evenly divisible by k.

**Example**  
 S = [19, 10, 12, 10, 24, 25, 22] k = 4

One of the arrays that can be created is S’[0] = [10, 12, 25]. Another is S’[1] = [19, 22, 24]. After testing all permutations, the maximum length solution array has 3 elements.

**Input Format**

The first line contains 2 space-separated integers, n and k, the number of values in S and the *non* factor.

The second line contains n space-separated integers, each an S[i], the unique values of the set.

**Constraints**

* All of the given numbers are distinct.

**Sample Input**

STDIN Function

----- --------

4 3 S[] size n = 4, k = 3

1 7 2 4 S = [1, 7, 2, 4]

**Sample Output**

3

**Explanation**

The sums of all permutations of two elements from S = {1, 7, 2, 4} are:

1 + 7 = 8

1 + 2 = 3

1 + 4 = 5

7 + 2 = 9

7 + 4 = 11

2 + 4 = 6

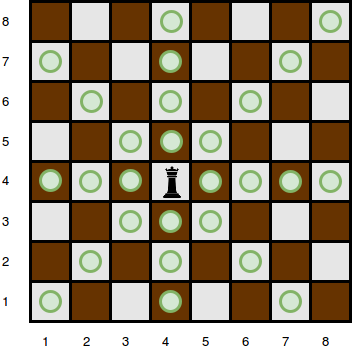
Only S’ = {1, 7, 4} will not ever sum to a multiple of k = 3.

* + 1. [25210016] **Queen’s Attack II**

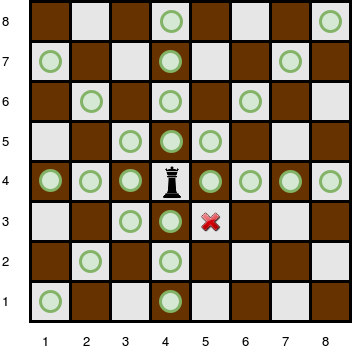
You will be given a square chess board with one queen and a number of obstacles placed on it. Determine how many squares the queen can attack.

A [queen](https://en.wikipedia.org/wiki/Queen_%28chess%29) is standing on an n × n [chessboard](https://en.wikipedia.org/wiki/Chess). The chess board's rows are numbered from 1 to n, going from bottom to top. Its columns are numbered from 1 to n, going from left to right. Each square is referenced by a tuple, (r, c), describing the row, r, and column, c, where the square is located.

The queen is standing at position . In a single move, she can attack any square in any of the eight directions (left, right, up, down, and the four diagonals). In the diagram below, the green circles denote all the cells the queen can attack from (4, 4):



There are obstacles on the chessboard, each preventing the queen from attacking any square beyond it on that path. For example, an obstacle at location (3, 5) in the diagram above prevents the queen from attacking cells (3, 5), (2, 6), and (1, 7):



Given the queen's position and the locations of all the obstacles, find and print the number of squares the queen can attack from her position at . In the board above, there are 24 such squares.

**Input Format**

The first line contains two space-separated integers n and k, the length of the board's sides and the number of obstacles.  
The next line contains two space-separated integers  and , the queen's row and column position.  
Each of the next k lines contains two space-separated integers  and , the row and column position of .

**Constraints**

* A single cell may contain more than one obstacle.
* There will never be an obstacle at the position where the queen is located.

**Subtasks**

For 30% of the maximum score:

For  of the maximum score:

**Sample Input 0**

4 0

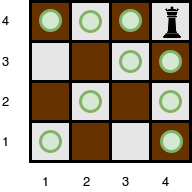
4 4

**Sample Output 0**

9

**Explanation 0**

The queen is standing at position (4, 4) on a 4 × 4 chessboard with no obstacles:



**Sample Input 1**

5 3

4 3

5 5

4 2

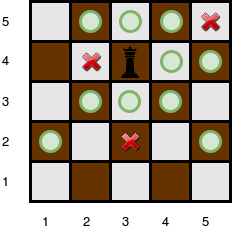
2 3

**Sample Output 1**

10

**Explanation 1**

The queen is standing at position (4, 3) on a 5 × 5 chessboard with k = 3 obstacles:



The number of squares she can attack from that position is .

**Sample Input 2**

1 0

1 1

**Sample Output 2**

0

**Explanation 2**

Since there is only one square, and the queen is on it, the queen can move 0 squares.

* + 1. [25210017] **ACM ICPC Team**

There are a number of people who will be attending [ACM-ICPC World Finals](https://en.wikipedia.org/wiki/ACM_International_Collegiate_Programming_Contest). Each of them may be well versed in a number of topics. Given a list of topics known by each attendee, presented as binary strings, determine the maximum number of topics a 2-person team can know. Each subject has a column in the binary string, and a '1' means the subject is known while '0' means it is not. Also determine the number of teams that know the maximum number of topics. Return an integer array with two elements. The first is the maximum number of topics known, and the second is the number of teams that know that number of topics.

**Example**

The attendee data is aligned for clarity below:

10101

11110

00010

These are all possible teams that can be formed:

Members Subjects

(1,2) [1,2,3,4,5]

(1,3) [1,3,4,5]

(2,3) [1,2,3,4]

In this case, the first team will know all 5 subjects. They are the only team that can be created that knows that many subjects, so [5, 1] is returned.

**Input Format**

The first line contains two space-separated integers n and m, where m is the number of attendees and  is the number of topics.

Each of the next n lines contains a binary string of length m.

**Constraints**

**Sample Input**

4 5

10101

11100

11010

00101

**Sample Output**

5

2

**Explanation**

Calculating topics known for all permutations of 2 attendees we get:

The 2 teams (1, 3) and (3, 4) know all 5 topics which is maximal.

* + 1. [25210018] **Organizing Containers of Balls**

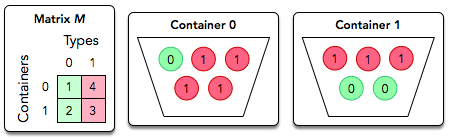
David has several containers, each with a number of balls in it. He has just enough containers to sort each type of ball he has into its own container. David wants to sort the balls using his sort method.

David wants to perform some number of swap operations such that:

* Each container contains only balls of the same type.
* No two balls of the same type are located in different containers.

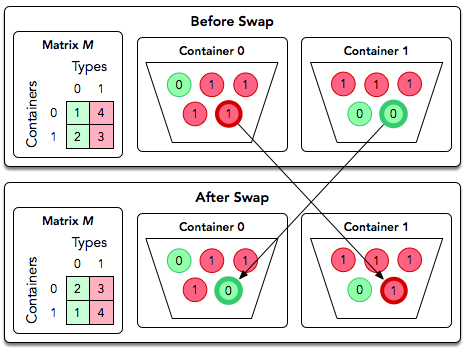
**Example**

David has n = 2 containers and 2 different types of balls, both of which are numbered from 0 to n – 1 = 1. The distribution of ball types per container are shown in the following diagram.



In a single operation, David can *swap* two balls located in different containers.

The diagram below depicts a single swap operation:



In this case, there is no way to have all green balls in one container and all red in the other using only swap operations. Return Impossible.

You must perform q queries where each query is in the form of a matrix, M. For each query, print Possible on a new line if David can satisfy the conditions above for the given matrix. Otherwise, print Impossible.

**Input Format**

The first line contains an integer q, the number of queries.

Each of the next q sets of lines is as follows:

1. The first line contains an integer n, the number of containers (rows) and ball types (columns).
2. Each of the next n lines contains n space-separated integers describing row containers[i].

**Constraints**

**Scoring**

* For 33% of score, .
* For 100% of score, .

**Output Format**

For each query, print Possible on a new line if David can satisfy the conditions above for the given matrix. Otherwise, print Impossible.

**Sample Input 0**

2

2

1 1

1 1

2

0 2

1 1

**Sample Output 0**

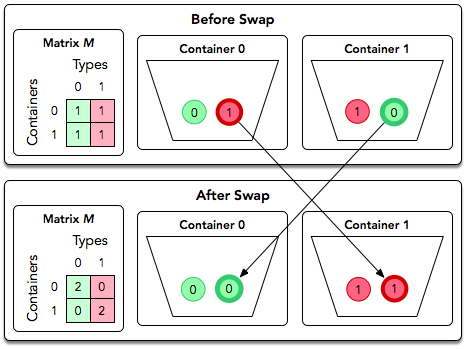
Possible

Impossible

**Explanation 0**

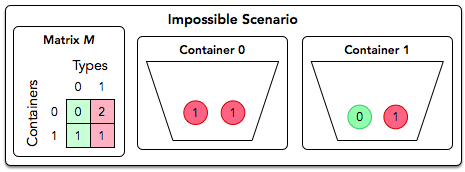
We perform the following q = 2 queries:

1. The diagram below depicts one possible way to satisfy David's requirements for the first query:



Thus, we print Possible on a new line.

1. The diagram below depicts the matrix for the second query:



No matter how many times we swap balls of type  and  between the two containers, we'll never end up with one container only containing type  and the other container only containing type . Thus, we print Impossible on a new line.

**Sample Input 1**

2

3

1 3 1

2 1 2

3 3 3

3

0 2 1

1 1 1

2 0 0

**Sample Output 1**

Impossible

Possible

* + 1. [25210019] **Encryption**

An English text needs to be encrypted using the following encryption scheme.

First, the spaces are removed from the text. Let L be the length of this text.

Then, characters are written into a grid, whose rows and columns have the following constraints:



**Example**



After removing spaces, the string is 54 characters long.  is between 7 and 8, so it is written in the form of a grid with 7 rows and 8 columns.

ifmanwas

meanttos

tayonthe

groundgo

dwouldha

vegivenu

sroots

* Ensure that
* If multiple grids satisfy the above conditions, choose the one with the minimum area, i.e.  .

The encoded message is obtained by displaying the characters of each column, with a space between column texts. The encoded message for the grid above is:

imtgdvs fearwer mayoogo anouuio ntnnlvt wttddes aohghn sseoau

Create a function to encode a message.

**Input Format**

One line of text, the string s

**Constraints**

 s contains characters in the range ascii[a-z] and space, ascii(32).

**Sample Input**

haveaniceday

**Sample Output 0**

hae and via ecy

**Explanation 0**

,  is between 3 and 4.

Rewritten with 3 rows and 4 columns:

have

anic

eday

**Sample Input 1**

feedthedog

**Sample Output 1**

fto ehg ee dd

**Explanation 1**

,  is between 3 and 4.

Rewritten with 3 rows and 4 columns:

feed

thed

og

**Sample Input 2**

chillout

**Sample Output 2**

clu hlt io

**Explanation 2**

,  is between 2 and 3.

Rewritten with 3 columns and 3 rows (2 × 3 = 6 < 8 so we have to use 3 × 3.)

chi

llo

ut

* + 1. [25210020] **Bigger is Greater**

[*Lexicographical order*](https://en.wikipedia.org/wiki/Lexicographical_order) is often known as alphabetical order when dealing with strings. A string is greater than another string if it comes later in a lexicographically sorted list.

Given a word, create a new word by swapping some or all of its characters. This new word must meet two criteria:

* It must be greater than the original word
* It must be the smallest word that meets the first condition

**Example**

The next largest word is abdc.

Complete the function biggerIsGreater below to create and return the new string meeting the criteria. If it is not possible, return no answer.

**Input Format**

The first line of input contains T, the number of test cases.

Each of the next T lines contains w.

**Constraints**

* w will contain only letters in the range ascii[a..z].

**Sample Input 0**

5

ab

bb

hefg

dhck

dkhc

**Sample Output 0**

ba

no answer

hegf

dhkc

hcdk

**Explanation 0**

* Test case 1:

ba is the only string which can be made by rearranging ab. It is greater.

* Test case 2:

It is not possible to rearrange bb and get a greater string.

* Test case 3:

hegf is the next string greater than hefg.

* Test case 4:

dhkc is the next string greater than dhck.

* Test case 5:

hcdk is the next string greater than dkhc.

**Sample Input 1**

6

lmno

dcba

dcbb

abdc

abcd

fedcbabcd

**Sample Output 1**

lmon

no answer

no answer

acbd

abdc

fedcbabdc

**Guide:**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

* + 1. [25210021] **Modified Kaprekar Numbers**

A modified Kaprekar number is a positive whole number with a special property. If you square it, then split the number into two integers and sum those integers, you have the same value you started with.

Consider a positive whole number n with d digits. We square n to arrive at a number that is either 2 × d digits long or (2 × d) – 1 digits long. Split the string representation of the square into two parts, l and r. The right hand part, r must be d digits long. The left is the remaining substring. Convert those two substrings back to integers, add them and see if you get n.

**Example**

n = 5

d = 1

First calculate that . Split that into two strings and convert them back to integers 2 and 5. Test 2 + 5 = 7 5, so this is not a modified Kaprekar number. If n = 9, still d = 1, and . This gives us 8 + 1 = 9, the original n.

**Note:** r may have leading zeros.

Here's an explanation from Wikipedia about the **ORIGINAL** [Kaprekar Number](https://en.wikipedia.org/wiki/Kaprekar_number) (spot the difference!):

In mathematics, a Kaprekar number for a given base is a non-negative integer, the representation of whose square in that base can be split into two parts that add up to the original number again. For instance, 45 is a Kaprekar number, because 45² = 2025 and 20 + 25 = 45.

Given two positive integers  and  where  is lower than , write a program to print the modified Kaprekar numbers in the range between  and , inclusive. If no modified Kaprekar numbers exist in the given range, print INVALID RANGE.

**Input Format**

The first line contains the lower integer limit p.

The second line contains the upper integer limit q.

**Note**: Your range should be inclusive of the limits.

**Constraints**

**Sample Input**

STDIN Function

----- --------

1 p = 1

100 q = 100

**Sample Output**

1 9 45 55 99

**Explanation**

1, 9, 45, 55, and 99 are the modified Kaprekar Numbers in the given range.

* + 1. [25210022] **Beautiful Triplets**

Given a sequence of integers a, a triplet  is beautiful if:

Given an increasing sequenc of integers and the value of d, count the number of beautiful triplets in the sequence.

**Example**

There are three beautiful triplets, by index: . To test the first triplet,  and .

**Input Format**

The first line contains 2 space-separated integers, n and d, the length of the sequence and the beautiful difference.  
The second line contains n space-separated integers .

**Constraints**

**Sample Input**

STDIN Function

----- --------

7 3 arr[] size n = 7, d = 3

1 2 4 5 7 8 10 arr = [1, 2, 4, 5, 7, 8, 10]

**Sample Output**

3

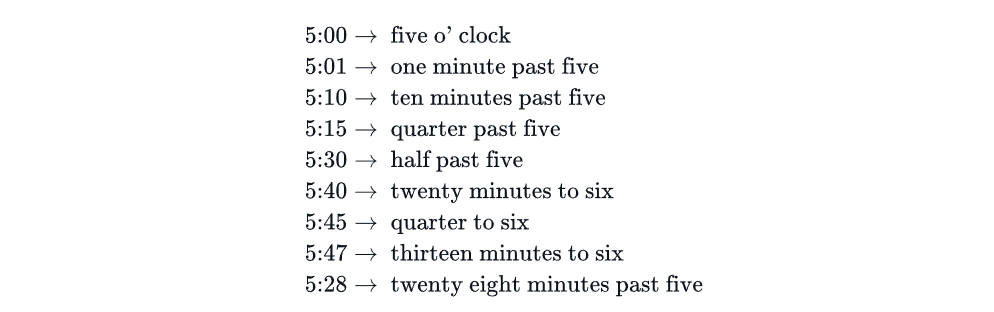
**Explanation**

There are many possible triplets , but our only beautiful triplets are  ,  and  by value, not index. Please see the equations below:

Recall that a beautiful triplet satisfies the following equivalence relation:  where .

* + 1. [25210023] **The Time in Words**

Given the time in numerals we may convert it into words, as shown below:



At minutes = 0, use *o' clock*. For , use *past*, and for  use *to*. Note the space between the apostrophe and *clock* in *o' clock*. Write a program which prints the time in words for the input given in the format described.

**Input Format**

The first line contains h, the hours portion The second line contains m, the minutes portion

**Constraints**

**Sample Input 0**

5

47

**Sample Output 0**

thirteen minutes to six

**Sample Input 1**

3

00

**Sample Output 1**

three o' clock

**Sample Input 2**

7

15

**Sample Output 2**

quarter past seven

* + 1. [25210024] **Halloween Sale**

You wish to buy video games from the famous online video game store Mist.

Usually, all games are sold at the same price, p dollars. However, they are planning to have the seasonal Halloween Sale next month in which you can buy games at a cheaper price. Specifically, the first game will cost p dollars, and every subsequent game will cost d dollars less than the previous one. This continues until the cost becomes less than or equal to  dollars, after which every game will cost m dollars. How many games can you buy during the Halloween Sale?

**Example**

The following are the costs of the first 11, in order:

Start at p = 20 units cost, reduce that by d = 3 units each iteration until reaching a minimum possible price, m = 6. Starting with s = 70 units of currency in your Mist wallet, you can buy 5 games: 20 + 17 + 14 + 11 + 8 = 70.

**Input Format**

The first and only line of input contains four space-separated integers p, d, m and s.

**Constraints**

**Sample Input 0**

20 3 6 80

**Sample Output 0**

6

**Explanation 0**

Assumptions other than starting funds, s, match the example in the problem statement. With a budget of 80, you can buy 6 games at a cost of 20 + 17 + 14 + 11 + 8 + 6 = 76. A 7th game for an additional 6 units exceeds the budget.

**Sample Input 1**

20 3 6 85

**Sample Output 1**

7

**Explanation 1**

This is the same as the previous case, except this time the starting budget s = 85 units of currency. This time, you can buy 7 games since they cost 20 + 17 + 14 + 11 + 8 + 6 + 6 = 82. An additional game at 6 units will exceed the budget.

* + 1. [25210025] **Lisa’s Workbook**

Lisa just got a new math workbook. A workbook contains exercise problems, grouped into chapters. Lisa believes a problem to be *special* if its index (within a chapter) is the same as the page number where it's located. The format of Lisa's book is as follows:

* There are n chapters in Lisa's workbook, numbered from 1 to n.
* The  chapter has arr[i] problems, numbered from 1 to arr[i].
* Each page can hold *up to* k problems. Only a chapter's last page of exercises may contain fewer than k problems.
* Each new chapter starts on a new page, so a page *will never* contain problems from more than one chapter.
* The page number indexing starts at 1.

Given the details for Lisa's workbook, can you count its number of *special* problems?

**Example**  
arr = [4, 2]

k = 3

Lisa's workbook contains arr[1] = 4 problems for chapter 1, and arr[2] = 2 problems for chapter 2. Each page can hold k = 3 problems.

The first page will hold 3 problems for chapter 1. Problem 1 is on page 1, so it is ***special*.** Page 2 contains only Chapter 1, Problem 4, so no *special* problem is on page 2. Chapter 2 problems start on page 3 and there are 2 problems. Since there is no problem 3 on page 3, there is no *special* problem on that page either. There is 1 *special* problem in her workbook.

**Note:** See the diagram in the *Explanation* section for more details.

**Input Format**

The first line contains two integers n and k, the number of chapters and the maximum number of problems per page.  
The second line contains n space-separated integers arr[i] where arr[i]  denotes the number of problems in the  chapter.

**Constraints**

**Sample Input**

STDIN Function

----- --------

5 3 n = 5, k = 3

4 2 6 1 10 arr = [4, 2, 6, 1, 10]

**Sample Output**

4

**Explanation**

The diagram below depicts Lisa's workbook with n = 5 chapters and a maximum of k = 3 problems per page. Special problems are outlined in red, and page numbers are in yellow squares.

Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, biểu đồ

Mô tả được tạo tự động

There are 4 special problems and thus we print the number 4 on a new line.