

Arrays

1. Largest Common End

Read **two arrays of words** and find the length of the **largest common end** (left or right).

Examples

Input	Output	Comments
hi php java csharp sql html css js hi php java js softuni nakov java learn	3	The largest common end is at the left: hi php java
hi php java xml csharp sql html css js nakov java sql html css js	4	The largest common end is at the right: sql html css js
I love programming Learn Java or C#	0	No common words at the left and right

Hints

- Scan the arrays from left to right until the end of the shorter is reached and count the equal elements.
- Scan the arrays from right to left until the start of the shorter is reached.
- Keep the start position and the length of the longest equal start / end.

2. Rotate and Sum

To “**rotate** an array on the right” means to move its last element first: {1, 2, 3} → {3, 1, 2}.

Write a program to read an array of **n integers** (space separated on a single line) and an integer **k**, rotate the array right **k times** and sum the obtained arrays after each rotation as shown below.

Examples

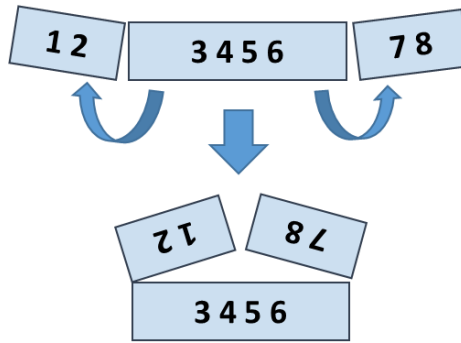
Input	Output	Comments
3 2 4 -1 2	3 2 5 6	rotated1[] = -1 3 2 4 rotated2[] = 4 -1 3 2 sum[] = 3 2 5 6
1 2 3 1	3 1 2	rotated1[] = 3 1 2 sum[] = 3 1 2
1 2 3 4 5 3	12 10 8 6 9	rotated1[] = 5 1 2 3 4 rotated2[] = 4 5 1 2 3 rotated3[] = 3 4 5 1 2 sum[] = 12 10 8 6 9

Hints

- After **r** rotations the element at position **i** goes to position **(i + r) % n**.
- The **sum[]** array can be calculated by two nested loops: for **r = 1 ... k**; for **i = 0 ... n-1**.

3. Fold and Sum

Read an array of **4*k integers**, fold it like shown below, and print the sum of the upper and lower two rows (each holding **2 * k integers**):



Examples

Input	Output	Comments
5 2 3 6	7 9	5 6 + 2 3 = 7 9
1 2 3 4 5 6 7 8	5 5 13 13	2 1 8 7 + 3 4 5 6 = 5 5 13 13
4 3 -1 2 5 0 1 9 8 6 7 -2	1 8 4 -1 16 14	-1 3 4 -2 7 6 + 2 5 0 1 9 8 = 1 8 4 -1 16 14

Hints

- Create the **first row** after folding: the first **k** numbers reversed, followed by the last **k** numbers reversed.
- Create the **second row** after folding: the middle **2*k** numbers.
- **Sum** the first and the second rows.

4. Sieve of Eratosthenes

Write a program to find **all prime numbers in range [1...n]**. Implement the algorithm called “Sieve of Eratosthenes”: https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes. Steps in the “Sieve of Eratosthenes” algorithm:

1. Assign **primes[0...n] = true**
2. Assign **primes[0] = primes[1] = false**
3. Find the smallest **p**, which holds **primes[p] = true**
 - Print **p** (it is prime)
 - Assign **primes[2*p] = primes[3*p] = primes[4*p] = ... = false**
4. Repeat for the next smallest **p < n**.

Examples

Input	Output
6	2 3 5
25	2 3 5 7 11 13 17 19 23

5. Compare Char Arrays

Compare two char arrays lexicographically (letter by letter).

Print the them in alphabetical order, each on separate line.

Examples

Input	Output
a b c d e f	abc def
p e t e r a n n i e	annie peter
a n n i e a n	an annie
a b a b	ab ab

Hints

- Compare the first letter of **arr1[]** and **arr2[]**, if equal, compare the next letter, etc.
- If all letters are equal, the smaller array is the **shorter**.
- If all letters are equal and the array lengths are the same, the arrays are **equal**.

6. Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

Examples

Input	Output
2 1 1 2 3 3 2 2 2 1	2 2 2
1 1 1 2 3 1 3 3	1 1 1
4 4 4 4	4 4 4 4
0 1 1 5 2 2 6 3 3	1 1

Hints

- Start with the sequence that consists of the first element: **start=0, len=1**.
- Scan the elements from left to right, starting at the second element: **pos=1...n-1**.
 - At each step compare the current element with the element on the left.
 - Same value → you have found a sequence longer by one → **len++**.
 - Different value → start a new sequence from the current element: **start=pos, len=1**.
 - After each step remember the sequence it is found to be longest at the moment:
bestStart=start, bestLen=len.
- Finally, print the longest sequence by using **bestStart** and **bestLen**.

7. Max Sequence of Increasing Elements

Write a program that finds the **longest increasing subsequence** in an array of integers. The longest increasing subsequence is a **portion of the array** (subsequence) that is strongly **increasing** and has the **longest possible length**. If several such subsequences exist, find the left most of them.

Examples

Input	Output
3 2 3 4 2 2 4	2 3 4
4 5 1 2 3 4 5	1 2 3 4 5
3 4 5 6	3 4 5 6
0 1 1 2 2 3 3	1 1

Hints

- Use the same algorithm like in the previous problem (Max Sequence of Equal Elements).

8. Most Frequent Number

Write a program that finds the **most frequent number** in a given sequence of numbers.

- Numbers will be in the range [0...65535].
- In case of multiple numbers with the same maximal frequency, print the leftmost of them.

Examples

Input	Output	Output
4 1 1 4 2 3 4 4 1 2 4 9 3	4	The number 4 is the most frequent (occurs 5 times)
2 2 2 2 1 2 2 2	2	The number 2 is the most frequent (occurs 7 times)
7 7 7 0 2 2 2 0 10 10 10	7	The numbers 2, 7 and 10 have the same maximal frequency (each occurs 3 times). The leftmost of them is 7.

9. Index of Letters

Write a program that creates an array containing all letters from the alphabet (a-z). Read a lowercase word from the console and print the **index of each of its letters in the letters array**.

Examples

Input	Output
abcz	a -> 0 b -> 1 c -> 2 z -> 25
softuni	s -> 18 o -> 14 f -> 5 t -> 19 u -> 20 n -> 13 i -> 8

10. Pairs by Difference

Write a program that **count the number of pairs** in given array **which difference is equal to given number**.

Input

- The **first line** holds the **sequence of numbers**.
- The **second line** holds the **difference**.

Examples

Input	Output	Comments
1 5 3 4 2 2	3	Pairs of elements with difference 2 -> {1, 3}, {5, 3}, {4, 2}
5 3 8 10 12 1 1	0	No pairs with difference 1

11. Equal Sums

Write a program that determines if there **exists an element in the array** such that the **sum of the elements on its left is equal to the sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **"no"** if there is no such index.

Examples

Input	Output	Comments
1 2 3 3	2	At a[2] -> left sum = 3, right sum = 3 a[0] + a[1] = a[3]
1 2	no	At a[0] -> left sum = 0, right sum = 2 At a[1] -> left sum = 1, right sum = 0 No such index exists
1	0	At a[0] -> left sum = 0, right sum = 0
1 2 3	no	No such index exists
10 5 5 99 3 4 2 5 1 1 4	3	At a[3] -> left sum = 20, right sum = 20 a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10]