Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 11: Dynamic Programming

Allowed languages: **C, C++**

General Tips

- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDE or text editors like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. However, in the lab you will be using command line interface.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- You can implement your solutions in C or C++. Using C++ STL (Standard Template Library) is allowed.
- When solving dynamic programming problems, try to think of your solution in terms of states (what does one state represent, how many total states do you need to cover all possible cases, etc.) and transitions (what are the valid transitions from a given state, what is the time complexity of each transition, etc.). This will make it easier for you to explain your solution and also analyze its time complexity (total time complexity of your solution would be #(states) * #(transitions from each state) * (time complexity of one transition)). Sometimes, checking the constraints can help you figure out if a dynamic programming solution to the problem would fit in the time limit and they might clue you as to what the states or transitions could be as well, so make sure you read the constraints of each problem carefully.

Problem A. Gymbag 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The BPHC gym has n dumbells, numbered $1, 2, 3, \ldots, n$. The i-th dumbbell has a weight of w_i and a value of v_i. Hurrshit has decided to choose some of the n dumbbells and carry them to the hostel in his gymbag. The capacity of his gymbag is w, which means that the sum of the weights of dumbbells taken must be at most w.

Find the maximum possible sum of the values of dumbbells that Hurrshit can take.

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(nk)$, where $k = \sum v_i$. Solution of higher time complexity will not be considered.

Input

The first line of input contains two space separated integers, n ($1 \leq n \leq 100$) and w ($1 \leq w \leq 10^9$) — the number of dumbbells and the capacity of Hurrshit's gymbag, respectively.

The following n lines contain the information about the dumbbells. The i-th line contains two space separated integers w_i ($1 \leq w_i \leq w$) and v_i ($1 \leq v_i \leq 10^3$) — the weight and value of the i-th dumbbell, respectively.

Output

Print one line containing the maximum possible sum of the values of dumbbells that Hurrshit can take.

Example

standard input	standard output

3 8
3 30
4 50
5 60
	90

1 1000000000 1000000000 10	10

Explanation

In example 1, dumbbells 1 and 3 should be taken. Then, the sum of the weights is $3 + 5 = 8$, and the sum of the values is $30 + 60 = 90$. It can be shown that no other set of dumbbells with sum of weights ≤ 8 can have sum of values > 90.

Problem B. Longest Common Subarray

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Given two integer arrays a and b of size n and m respectively, find the maximum length of a subarray that appears in both arrays.

Note:

A subarray is a contiguous non-empty sequence of elements within an array.

Assume 1-based indexing.

The time complexity of your solution should be $O(nm)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n $(1 \le n \le 10^3)$ — the size of array a.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ $(-10^9 \le a_i \le 10^9)$ — the elements of array a.

The third line of input contains m $(1 \le m \le 10^3)$ — the size of array b.

The fourth line of input contains m space separated integers $b_1, b_2, \ldots, b_m$ $(-10^9 \le b_i \le 10^9)$ — the elements of array b.

Output

Print one line containing the maximum length of a subarray that appears in both a and b. If there is no subarray that appears in both a and b, print 0.

Examples

standard input	standard output

5
1 2 3 2 1
6
3 2 1 4 7 -4
	3

3
1 2 3
2
4 5
	0

Explanation

In example 1, the repeated subarray with maximum length is $[3, 2, 1]$ and its length is 3.

In example 2, there is no subarray that appears in both a and b so we print 0.

Problem C. Edit Distance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Given two strings s and t of length n and m respectively, find the minimum number of operations required to convert s to t.

You have the following three operations permitted on a string:

- Insert a character
- Delete a character
- Replace a character

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(nm)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \le n \le 10^3$) — the length of string s.

The second line of input contains the string s.

The third line of input contains m ($1 \le m \le 10^3$) — the length of string t.

The fourth line of input contains the string t.

It is guaranteed that s and t consist of lowercase English letters only.

Output

Print one line containing the minimum number of operations required to convert s to t.

Examples

standard input	standard output

`5`
`horse`
`3`
`ros`
	`3`

`3`
`abc`
`4`
`defg`
	`4`

Explanation

In example 1, the following sequence of operations can convert s to t:

- horse → rorse (replace 'h' with 'r')
- rorse → rose (remove 'r')
- rose → ros (remove 'e')

In example 2, the following sequence of operations can convert s to t:

- abc $\rightarrow$ dbc (replace ‘a’ with ‘d’)
- dbc $\rightarrow$ dec (replace ‘b’ with ‘e’)
- dec $\rightarrow$ def (replace ‘c’ with ‘f’)
- def $\rightarrow$ defg (insert ‘g’)

Problem D. Our Blessings are also with You

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array a of n integers. Two players are playing a game with this array: Vidyateja and Hom. Note that they can see all the elements of array a.

Vidyateja and Hom take turns, with Vidyateja starting first. Both players start the game with a score of 0. At each turn, the player takes one of the numbers from either end of the array (i.e., `a[0]` or `a[n - 1]`) which reduces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.

At the end of the game, the player with the higher score is the winner. If both Vidyateja and Hom have the same score, then Vidyateja is the winner. Assume that both the players are playing optimally. Find the winner of the game.

Note:

Assume 0-based indexing.

The time complexity of your solution should be $O(n^2)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^3$) — the size of array a.

The second line of input contains n space separated integers $a_0, a_1, \ldots, a_{n-1}$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

Output

Print one line containing the name of the winner of the game.

Examples

standard input	standard output
3 1 5 2	Hom
4 1 5 233 7	Vidyateja

Explanation

In example 1, Vidyateja can initially choose between 1 and 2. If he chooses 2 (or 1), then Hom can choose from 1 (or 2) and 5. If Hom chooses 5, then Vidyateja will be left with 1 (or 2). So, the final score of Vidyateja is $1 + 2 = 3$, and Hom is 5. Hence, Vidyateja will never be the winner.

In example 2, Vidyateja can choose 1 first. Then Hom has to choose between 5 and 7. No matter which number Hom chooses, Vidyateja can choose 233. So, the final score of Vidyateja is 234, and Hom is 12.

Problem E. Matrix Chain Multiplication

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Let us consider the number of ordinary multiplication operations required for the multiplication of an $n \times m$ matrix with an $m \times k$ matrix. This is a 7th class mathematics problem, whose solution is $n \times m \times k$ operations.

As a domain expert with experience of 20 years, you are expected to solve the following more complex problem: Given a sequence of n matrices, find the most efficient way of multiplying them. The most efficient way is the one that minimizes the total number of ordinary multiplication operations.

You need not find the most optimal way of parenthesizing the sequence, finding the minimum number of ordinary multiplication operations required to find the product of the given sequence of n matrices is enough.

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(n^3)$. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($2 \leq n \leq 100$) — the number of matrices to be multiplied.

The second line of input contains $n + 1$ space separated integers $a_1, a_2, \ldots, a_{n+1}$ ($1 \leq a_i \leq 500$) — the i-th matrix of the sequence is an $a_i \times a_{i+1}$ matrix.

Output

Print one line containing the minimum number of ordinary multiplication operations that need to be performed.

Example

standard input	standard output

`3`
`10 30 5 60`
	`4500`

Explanation

Let the matrices of the sequence be A, B and C. The dimensions of A are 10×30, B are 30×5 and C are 5×60. Computing (AB)C needs $(10 \times 30 \times 5) + (10 \times 5 \times 60) = 1500 + 3000 = 4500$ operations while computing A(BC) needs $(30 \times 5 \times 60) + (10 \times 30 \times 60) = 9000 + 18000 = 27000$ operations. Clearly, the first method is more efficient.

Problem F. Coupon Change

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

It's the TechWeek on campus and your favourite food vendors are back. You don't want to overspend on your coupons so you look around the food stalls and decide to buy food worth k. The coupons come in n different dimensions - $c_1, c_2, c_3 \dots c_n$. You can buy a coupon worth c_i any number of times. Find the minimum number of coupons you need such that they are worth k.

Note that if it is not possible to get exactly k using the given coupon dimensions, print -1.

Note:

The time complexity of your solution should be $O(nk)$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of two integers n $(1 \leq n \leq 1000)$ and k $(1 \leq k \leq 1000)$ — the number of different dimension coupons and the amount to be spent on food.

The second line of input contains n space-separated distinct integers $c_1, c_2, \dots, c_n$ $(0 \leq c_i \leq 1000)$ — each integer represents the value of each coupon.

Output

The minimum number of coupons you need to get a total sum of k.

Examples

`standard input`	`standard output`

`3 11`
`3 4 5`
	`3`

`4 3`
`4 7 9 5`
	`-1`

Explanation

In the first example 11 can be obtained using 3 + 4 + 4, using 3 coupons. You cannot get 11 with lesser than 3 coupons.

In the second example, we cannot get exactly 3 since all coupons have value greater than 3.

Problem G. SubPalindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Alice is given a string s of length n. Alice likes patterns and palindromes in particular. She wants to know the number of substrings of s that are palindromes. Since this can be a very tedious task, she needs your help in doing it. Help Alice by finding the number of substrings that are palindromes.

Note:

A string is a palindrome when it reads the same backward as forward. A single letter is also considered a palindrome.

A substring is a contiguous sequence of characters within the string.

Assume 0-based indexing

The time complexity of your solution should be $O(n^2)$. Any solution with a higher time complexity will not be considered.

Input

The input consists of a single string, s $(1 \leq |s| \leq 1000)$.

Output

Print a single integer — the number of substrings that are palindromes.

Examples

standard input	standard output
`abcb`	`5`
`abbba`	`9`

Explanation

In the first example the substrings are - a (0,0), b (1,1), c (2,2), b (3,3), bcb (1,3)

In the second example the substring are - a (0,0), b (1,1), b (2,2), b (3,3), a (4,4), bb (1,2), bb (2,3), bbb (1,3), $abbba$ (0,4).

Problem H. RBS II

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

A bracket sequence is any sequence of '(' and ')'. For example ()(), (()),)(are all bracket sequences.

A bracket sequence is said to be **regular** if it is possible to obtain correct arithmetic expression by inserting characters + and 1 into this sequence. For example, sequences (())(), () and (()(())) are regular, while)(, (() and (()))(are not. A bracket sequence that is regular is called a Regular Bracket Sequence (RBS)

You are given a bracket sequence in the form of a string s, find the length of the longest substring of s that is a RBS.

Note:

The time complexity of your solution should be $O(n)$. Any solution with a higher time complexity will not be considered.

Input

The input consists of a single string, s ($1 \leq |s| \leq 10^6$) — the bracket sequence.

Output

Print the length of the longest substring of s that is a RBS.

Examples

standard input	standard output
`)()())`	`4`
`)(`	`0`

Explanation

In the first example the longest substring that is a RBS is ()()(1,4 assuming 0-based indexing)

In the second example no substring is a RBS.

Problem I. Minimize or Maximize

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Rahul and Kiran are bored and decide to play a new game. The game works as follows - They are initially given a sequence $a = [a_1, a_2, a_3 \dots a_n]$ of integers. Until a becomes empty, the two players play alternatively starting from Rahul. During their turn, each player removes an element either from the beginning or from the end of a and earns x points more where x is the element removed.

Let X and Y be Rahul's and Kiran's total score at the end of the game, respectively. At any point of time, Rahul tries to maximize $X - Y$, while Kiran tries to minimize $X - Y$.

Assume that the two players play optimally and find the value of $X - Y$.

Note:

The time complexity of your solution should be $O(n^2)$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of a single integer n ($1 \leq n \leq 1000$) — the number of elements in the initial sequence a.

The second line of input contains n space-separated integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$) — the elements of a

Output

Print the value of $X - Y$ if both players play optimally.

Examples

standard input	standard output

`4`
`10 80 90 30`
	`10`

`3`
`10 100 10`
	`-80`

Explanation

In the first example, (the element removed is written in bold)

1. Rahul: (10, 80, 90, **30**) $\rightarrow$ (10, 80, 90)
2. Kiran: (10, 80, **90**) $\rightarrow$ (10, 80)
3. Rahul: (10, **80**) $\rightarrow$ (10)
4. Kiran: (**10**) $\rightarrow$ ()

Here $X = 30 + 80 = 110$ and $Y = 90 + 10 = 100$. So $X - Y = 10$

In the second example,

1. Rahul: (**10**, 100, 10) $\rightarrow$ (100, 10)
2. Kiran: (**100**, 10) $\rightarrow$ (10)
3. Rahul: (**10**) $\rightarrow$ ()

Here $X = 20$ and $Y = 100$, so $X - Y = 20 - 100 = -80$

Problem J. CheckSum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array of integers of length n and an integer k. Check if there exists a subsequence of the array such that the sum of the elements is exactly k.

A subsequence of an array is a sequence that can be derived from the array by removing zero or more elements from it.

Note:

The time complexity of your solution should be $O(nk)$. Any solution with a higher time complexity will not be considered.

Input

The first line consists of two integers n ($1 \leq n \leq 1000$) and k ($1 \leq k \leq 5000$) — the number of elements in the array and the sum to be obtained.

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^5$) — the elements of the array.

Output

Print Yes if there exists a subsequence with sum k, otherwise print No.

Example

standard input	standard output

`5 15`
`3 1 5 9 12`
	`Yes`

`4 4`
`7 2 3 5`
	`No`

Explanation

In the first example, we can obtain 15 by taking the subsequence [3 12]. We can also get 15 from the subsequence [1 5 9]

In the second example, no matter what subsequence we take, we cannot get 4.