A. Wrong Subtraction

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

Little girl Tanya is learning how to decrease a number by one, but she does it wrong with a number consisting of two or more digits. Tanya subtracts one from a number by the following algorithm:

- if the last digit of the number is non-zero, she decreases the number by one;
- if the last digit of the number is zero, she divides the number by 10 (i.e. removes the last digit).

You are given an integer number n. Tanya will subtract one from it k times. Your task is to print the result after all k subtractions.

It is guaranteed that the result will be positive integer number.

Input

The first line of the input contains two integer numbers n and k ($2 \le n \le 10^9$, $1 \le k \le 50$) — the number from which Tanya will subtract and the number of subtractions correspondingly.

Output

Print one integer number — the result of the decreasing n by one k times.

It is guaranteed that the result will be positive integer number.

Examples

input
512 4

output
50

input
1000000000 9

output
1

Note

The first example corresponds to the following sequence: $512 \to 511 \to 510 \to 51 \to 50$.

A. Football

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

Petya loves football very much. One day, as he was watching a football match, he was writing the players' current positions on a piece of paper. To simplify the situation he depicted it as a string consisting of zeroes and ones. A zero corresponds to players of one team; a one corresponds to players of another team. If there are at least 7 players of some team standing one after another, then the situation is considered dangerous. For example, the situation 00100110111111101 is dangerous and 11110111011101 is not. You are given the current situation. Determine whether it is dangerous or not.

Input
The first input line contains a non-empty string consisting of characters "0" and "1", which represents players. The length of the string does not exceed 100 characters. There's at least one player from each team present on the field.

Output
Print "YES" if the situation is dangerous. Otherwise, print "NO".

Examples

input	Copy
001001	
output	Copy
NO	

input	Copy
1000000001	
output	Copy
YES	

A. Soldier and Bananas

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

A soldier wants to buy w bananas in the shop. He has to pay k dollars for the first banana, $2k$ dollars for the second one and so on (in other words, he has to pay $i \cdot k$ dollars for the i-th banana).

He has n dollars. How many dollars does he have to borrow from his friend soldier to buy w bananas?

Input
The first line contains three positive integers k, n, w ($1 \leq k, w \leq 1000$, $0 \leq n \leq 10^9$), the cost of the first banana, initial number of dollars the soldier has and number of bananas he wants.

Output
Output one integer — the amount of dollars that the soldier must borrow from his friend. If he doesn't have to borrow money, output 0.

Examples

input	Copy
3 17 4	
output	Copy
13	

Christmas Tree Pattern
Write a program that takes an integer as input and prints a Christmas tree pattern with the given number of levels. For example, if the input is 5, the program should print a Christmas tree like this:

Hollow Diamond Pattern
Write a program that takes an odd integer as input and prints a hollow diamond pattern with the given number of rows. For example, if the input is 7, the program should print a pattern like this:

Pyramid Pattern
Write a program that takes an integer as input and prints a pyramid pattern with the given number of levels. For example, if the input is 4, the program should print a pyramid like this:

Inverted Pyramid Pattern
Write a program that takes an integer as input and prints an inverted pyramid pattern with the given number of levels. For example, if the input is 3, the program should print an inverted pyramid like this:

Square Pattern
Write a program that takes an integer as input and prints a square pattern with the given number of rows and columns. For example, if the input is 5, the program should print a square like this:

1. **Find Maximum Element:**
   Given a 1D array, write a program to find the maximum element in the array.

2. **Find Minimum Element:**
   Write a program to find the minimum element in a 1D array.

3. **Sum of Elements:**
   Calculate the sum of all elements in a 1D array.

4. **Find the Second Largest Element:**
   Write a program to find the second largest element in a 1D array.

5. **Reverse the Array:**
   Given a 1D array, write a program to reverse the elements in the array. For example, if you have `[1, 2, 3, 4, 5]`, the program should transform it into `[5, 4, 3, 2, 1]`.

6. **Remove Duplicates:**
   Write a program to remove duplicates from a 1D array and return a new array with unique elements.

7. **Count Occurrences:**
   Given an element, write a program to count the number of times it occurs in a 1D array.

8. **Sort the Array:**
   Implement a sorting algorithm (e.g., bubble sort, insertion sort) to sort the elements in a 1D array in ascending or descending order.

9. **Find Subarray with Maximum Sum:**
   Write a program to find the subarray (contiguous elements) with the maximum sum in a 1D array. This is also known as the "maximum subarray problem."

10. **Find the Missing Number:**
    Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, write a program to find the missing number in the sequence.

1. **Print Horizontal Rows:**
   Given a 2D matrix (a list of lists), write a program to print all the horizontal rows. For example, if you have the following 2D matrix:
   ```
   1 2 3
   4 5 6
   7 8 9
   ```
   The program should print:
   ```
   1 2 3
   4 5 6
   7 8 9
   ```

2. **Print Vertical Columns:**
   Write a program to print all the vertical columns of a given 2D matrix. Using the same matrix as above, the program should print:
   ```
   1 4 7
   2 5 8
   3 6 9
   ```

3. **Find Maximum Element:**
   Given a 2D matrix, write a program to find the maximum element in the matrix.

4. **Sum of Elements:**
   Write a program to calculate the sum of all elements in a 2D matrix.

5. **Transpose a Matrix:**
   Write a program to transpose a given 2D matrix. This means swapping rows and columns. For example, the transpose of the matrix given above would be:
   ```
   1 4 7
   2 5 8
   3 6 9
   ```

6. **Rotate a Matrix 90 Degrees:**
   Given a 2D matrix, write a program to rotate it 90 degrees clockwise. For example, if you have the following matrix:
   ```
   1 2 3
   4 5 6
   7 8 9
   ```
   The program should rotate it to:
   ```
   ```

```
7 4 1
8 5 2
9 6 3
```

7. **Diagonal Elements:**
   Write a program to print the diagonal elements of a 2D matrix. For the matrix given above, the program should print: `1 5 9`.