Exercise: Dynamic Programming

This document defines the lab for the "Algorithms – Fundamentals (Java)" course @ Software University.

Please submit your solutions (source code) to all below-described problems in Judge.

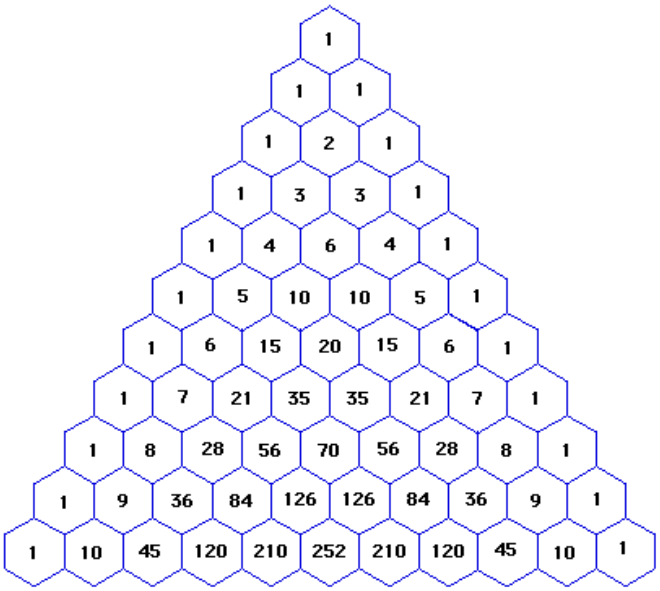
1. Binomial Coefficients

Write a program that finds the binomial coefficient $\binom{n}{k}$ for given non-negative integers **n** and **k**. The coefficient can be found recursively by adding the two numbers above using the formula:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{for all integers } n, k : 1 \leq k \leq n-1,$$

However, this leads to calculating the same coefficient multiple times (a problem that also occurs when solving the Fibonacci problem recursively). Use memoization to improve performance.

You can check your answers using the picture below (row and column indices start from 0):



Examples

Input	Output
3 2	3
4 0	1
6 2	15
10 5	252

2. Longest Zigzag Subsequence

A zigzag sequence is one that alternately increases and decreases. More formally, such a sequence has to comply with one of the two rules below:

1) Every even element is smaller than its neighbors and every odd element is larger than its neighbors, or
2) Every odd element is smaller than its neighbors and every even element is larger than its neighbors

1 3 2 is a zigzag sequence, but 1 2 3 is not. Any sequence of one or two elements is zig zag.

Find the longest zigzag subsequence in a given sequence.

Examples

Input	Output
8 3 5 7 0 8 9 10 20 20 20 12 19 11	8 3 5 0 20 12 19 11
1 2 3	1 2
1 3 2	1 3 2
24 5 31 3 3 342 51 114 52 55 56 58	24 5 31 3 342 51 114 52 55

3. Dividing Presents

Alan and Bob are twins. For their birthday they received some presents and now they need to split them amongst themselves. The goal is to **minimize the difference between the values of the presents received by the two brothers**, i.e. to divide the presents as equally as possible.

Assume the presents have values represented by positive integer numbers and that presents cannot be split in half (a present can only go to one brother or the other).

Find the minimal difference that can be obtained and print which presents each brother has received (you may only print the presents for one of them, it is obvious that the rest will go to the other brother). In the examples below Alan always takes a value less than or equal to Bob, but you may do it the other way around.

Examples

Input	Output
3 2 3 2 2 77 89 23 90 11	Difference: 30 Alan:136 Bob:166 Alan takes: 11 90 23 2 2 3 2 3 Bob takes the rest.
2 2 4 4 1 1	Difference: 0 Alan:7 Bob:7 Alan takes: 1 4 2 Bob takes the rest.

7 17 45 91 11 32 102 33 6 3	Difference: 1 Alan:173 Bob:174 Alan takes: 33 32 91 17 Bob takes the rest.
1 1 1 1 1 1 1 1 1 22	Difference: 13 Alan:9 Bob:22 Alan takes: 1 1 1 1 1 1 1 1 1 Bob takes the rest.

4. Sum with Unlimited Amount of Coins

We have a set of coins with predetermined values, e.g. 1, 2, 5, 10, 20, 50. Given a sum of **S**, the task is to find how many combinations of coins will sum up to **S**. For each value, we can use an **unlimited number of coins**, e.g. we can use S coins of value 1 or S/2 coins of value 2 (if S is even), etc.

Examples

Input	Output	Comments
1 2 3 4 6 6	10	The 10 combinations are: 6 = 6 6 = 4 + 2 6 = 4 + 1 + 1 6 = 3 + 3 6 = 3 + 2 + 1 6 = 3 + 1 + 1 + 1 6 = 2 + 2 + 2 6 = 2 + 2 + 1 + 1 6 = 2 + 1 + 1 + 1 + 1 6 = 1 + 1 + 1 + 1 + 1 + 1
1 2 5 5	4	The 4 combinations are: 5 = 5 5 = 2 + 2 + 1 5 = 2 + 1 + 1 + 1 5 = 1 + 1 + 1 + 1 + 1
1 2 5 10 13	16	
1 2 5 10 20 50 100	4563	

100		

5. Sum with Limited Amount of Coins

In the previous problem, the coins represented values, not actual coins (we could take as many coins of a certain value as we wanted). In this problem, we'll regard the coins as actual coins, e.g. 1, 2, and 5 are three coins and we can use each of them only once. We can, of course, have more coins of a given value, e.g. – 1, 1, 2, 2, 10.

The task is the same - find the number of ways we can combine the coins to obtain a certain sum **S**.

Examples

Input	Output	Comments
1 2 2 3 3 4 6 6	4	The 4 combinations are: 6 = 6 6 = 4 + 2 6 = 3 + 3 6 = 3 + 2 + 1
1 2 2 5 5	2	The 2 combinations are: 5 = 5 5 = 2 + 2 + 1
1 2 2 5 5 10 13	2	The 2 combinations are: 13 = 10 + 2 + 1 13 = 5 + 5 + 2 + 1
50 20 20 20 20 20 10 100	2	The 2 combinations are: 100 = 50 + 20 + 20 + 10 100 = 20 + 20 + 20 + 20 + 20

6. Word Differences

Write a program that finds all the **differences** in two strings with **equal lengths**. You have to determine the **smallest set** of **deletions** and **insertions** to make the **first** string **equal** to the **second**. Finally, you have to print the **count** of the minimum **insertions** and **deletions**.

Input

- You will receive the **two strings** on **separate lines.**

Output

- Print the minimum **amount** of **deletions** and **insertions** as described below.

Examples

Input	Output	Comment
YMCA HMBB	Deletions and Insertions: 6	One solution will be to remove "Y" and add "H" to the first: HMCA "M" matches in both strings Remove "C" and "A" from the first: HM Add two "B"'s and now both strings match
JFEIOWHGOW GHEWQHFEWQ	Deletions and Insertions: 12	

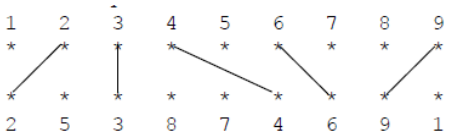
7. Connecting Cables

We are in a rectangular room. On opposite sides of the room, there are sets of **n** cables (n < 1000). The cables are indexed from 1 to n.

On each side of the room, there is a permutation of the cables, e.g. on one side we always have ordered {1, 2, 3, 4, 5} and on the other side, we have some permutation {5, 1, 3, 4, 2}. We are trying to connect each cable from one side with the corresponding cable on the other side – connect 1 with 1, 2 with 2, etc. **The cables are straight and should not overlap!**

The task is to find the maximum number of pairs we can connect given the restrictions above.

Examples

Input	Output	Comments
2 5 3 8 7 4 6 9 1	Maximum pairs connected: 5	
4 3 2 1	Maximum pairs connected: 1	Any other pair can be connected as well.
1 2 3	Maximum pairs connected: 3	

8. Minimum Edit Distance

We have two strings, **s1** and **s2**. The goal is to obtain **s2** from **s1** by applying the following operations:

- **replace(i, x)** – in **s1**, replace the symbol at index **i** with the character **x**
- **insert(i, x)** – in **s1**, inserts the character **x** at index **i**
- **delete(i)** – from **s1**, remove the character at index **i**

We are only allowed to modify **s1**, **s2** stays unchanged at all times. Each of the three operations has a certain **cost** associated with it (positive integer number). **Note**: the cost of the **replace(i, x)** operation is 0 if it doesn't actually change the character.

The goal is to find the sequence of operations which will produce **s2** from **s1** with **minimal cost**.

Input

- The input consists of **five lines**, on the **first** line the **cost** for **replace** on the **second** the **cost** for **insert,** and then on the **third** the **cost** for the **delete**. After that on the next two lines are the two strings **s1 and s2.**

Examples

Input	Output	Comments
3 2 1 abracadabra mabragabra	Minimum edit distance: 7	Indices refer to the original s1 string – DELETE(3) deletes the symbol at index 3 from abracadabra, not from the modified string mabracadabra after the INSERT(0, m) operation.
5 2 1 nqma bira ima bira	Minimum edit distance: 4	We can obtain s2 with two operations – DELETE(0) + REPLACE(1, i), but the cost of the REPLACE operation is high, that's why the solution involves three operations, their total cost is smaller. The INSERT can be performed also at index 0 and index 2.
3 3 3 equal equal	Minimum edit distance: 0	
1 1 1 equal different	Minimum edit distance: 8	