Each exercise is 2 points

3.1 Compare Sums

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given two arrays of floats A and B, check which one has a larger sum of elements.

Input

The first line contains an integer n ($1 \leq n \leq 100$), the length of the arrays.

The second line contains floating point numbers $A[1], A[2], \ldots, A[n]$ ($0.001 \leq A[i] \leq 1\,000$, for all $1 \leq i \leq n$), the elements of A.

The third line contains floating point numbers $B[1], B[2], \ldots, B[n]$ ($0.001 \leq B[i] \leq 1\,000$, for all $1 \leq i \leq n$), the elements of B.

Each floating point number has exactly 3 digits after the decimal point.

Output

Output `SUM(A)=SUM(B)`, if both sequences have equal sum of elements. Output `SUM(A)>SUM(B)`, if sequence A has larger sum. Otherwise output `SUM(A)<SUM(B)`. Avoid any white spaces.

Examples

standard input	standard output
2 1.500 1.500 1.000 2.000	SUM(A)=SUM(B)
1 2.000 1.123	SUM(A)>SUM(B)
3 1.000 2.000 3.000 1.001 2.001 3.001	SUM(A)<SUM(B)

3.2 Round Up

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a rational number $\frac{x}{y}$, find the minimum integer z such that $z \geq \frac{x}{y}$.

Input

Two integers x and y ($-10^9 \leq x, y \leq 10^9$, $y \neq 0$).

Output

The required value z.

Examples

standard input	standard output
1 2	1
1 -2	0
10 10	1

3.3 Yet Another Sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a sequence of integers $x_1, x_2, \ldots, x_n$, compute the sum

$$s = \sum_{i=1}^{n} \left(x_i + \frac{1}{x_i} \right) .$$

Pay attention to the precision of the resulting *double*.

Input

The first line contains an integer n ($1 \le n \le 50$), the length of the sequence.

The second line contains integers $x_1, x_2, \ldots, x_n$ ($-10^9 \le x_i \le 10^9$, $x_i \neq 0$, for all $1 \le i \le n$).

Output

A floating-point number s. The answer will be graded as correct, if absolute or relative error does not exceed 10^{-9}.

Examples

standard input	standard output
1 3	3.3333333333
1 -4	-4.2500000000
4 -2 -3 1 2	-1.3333333333

3.4 Binary Knapsack

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

The *knapsack problem* is a classical problem in combinatorial optimization: Given a set of n items, each with a weight w_i and a value v_i, determine the items to include in a collection so that the total weight is less than or equal to a given limit w and the total value is as large as possible.

Your goal is to solve a special case of the knapsack problem with additional constraint: the weight of each item is a power of 2.

Find the the set of items with the maximum total value so that their total weight is at most w.

Input

The first line contains two integers n and w ($1 \le n \le 100$, $0 \le w \le 2^{30}$), the number of items and the maximal total weight of items in the knapsack.

Each of the next n lines contains two integers w_i and v_i ($1 \le w_i \le 2^{30}$, $0 \le v_i \le 10^6$), the weight and the value of i-th item in the set. Each w_i is a power of 2.

Output

The maximum total value of items in the selected set.

Examples

standard input	standard output
3 5 1 3 1 2 1 1	6
5 10 1 4 2 5 1 2 4 6 8 12	18
1 5 8 3	0

Note

Each item can be selected at most once.