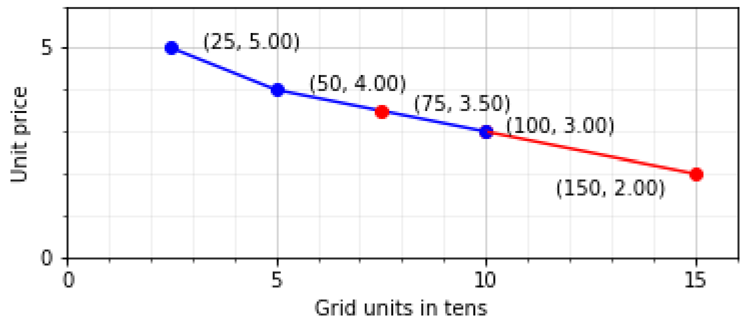
8. VM Pricing

A cloud service provider offers quantity discounts based on the number of virtual machines a customer needs. Their offerings vary from *2* to *2000* instances. When pricing is requested, the customer's representative refers to a list of past pricing. Given a list of past price quotes and the number of instances a customer needs, determine the per-instance price for the customer.

The method for determining price is as follows:

- If the number of instances needed is *exactly the same* as the quantity for a prior customer, the unit price is that price.
- If there is a price for a larger number and a price for a smaller number of instances, linearly interpolate the price of the quantity needed from the unit prices for the closest smaller and larger quantities.
- If the quantities for which there is past data are *all smaller* or *all larger* than the amount needed, then linearly extrapolate the unit price from the *2* points *closest* to the quantity needed.
- If the database only has *1* quantity, that is the price per unit.
- Sometimes, price quotes lapse. When that happens, the old pricing is overwritten with either a *0* or a negative number. The quantities associated with zero or negative unit prices must be disregarded.

For example, assume the price breaks occur for *instances = [25, 50, 100]* units at *price = [5.0, 4.0, 3.0]*. A diagram follows with pricing for *75* and *150* units. In the graph, price versus quantity for given values are in blue. The target numbers of instances and the linear extrapolation are plotted in red.



Function Description

Complete the function *interpolate* in the editor below. The function must return the expected price per unit rounded to two places after the decimals and cast as a string.

interpolate has the following parameter(s):

n: an integer that denotes the number of instances required *instances[instances[0],...instances[m-1]]:* an array of integers in increasing order, each a number of instances ordered in the past

price[price[0],...price[m-1]]: an array of floating point numbers where *price[i]* is the unit price when *instances[i]* units were purchased.

Note: The *interpolate* function's array parameters may be vectors, where necessitated by the language.

Constraints

- $2 \leq n \leq 2000$
- $1 \leq m \leq 100$
- $|instances| = |price| = m$
- *instances[i]* < *instances[j]*, where $0 \leq i < j < m$

Input Format For Custom Testing

The first line contains an integer, *n*, the number of instances needed.

The second line contains an integer, *m*, the size of the array *instances*.

Each line *i* of the *m* subsequent lines (where $0 \leq i < m$) contains an integer that describes *instances[i]*.

The next line again contains the integer, *m*, the size of the array price.

Each line *i* of the *m* subsequent lines (where $0 \leq i < m$) contains a floating point number that describes *price[i]*.

Sample Case 0

Sample Input 0

```
25
5
10
25
```

```
50
100
500
5
2.46
2.58
2.0
2.25
3.0
```

Sample Output 0

```
2.58
```

Explanation 0

The following arguments are passed to the *interpolate* function:

n = 25

instances = { 10, 25, 50, 100, 500 }

price = { 2.46, 2.58, 2.0, 2.25, 3.0 }

The quantity *25* is in the database, so *vmPricing* returns the
unit price associated with that quantity, *2.58*, cast as a string.

Sample Case 1