Problem A: Feng Shui

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are currently moving into a new apartment. Naturally, before unpacking anything, you want to know the apartment's Feng Shui, that is, the maximum Feng Shui of any location in it. Fortunately, the landlord already gave you all the information necessary for determining it: When the apartment's groundplan is written as a simple polygon in the right coordinates, the Feng Shui at any given coordinates in the property can be described as the dot product of the point with these coordinates and the apartment's Feng Shui vector – and the landlord kindly provided both the polygon-model of the apartment and the Feng Shui vector itself to you.

Input

The input consists of

- one line containing integers v_x and v_y ($-10^9 \leq v_x, v_y \leq 10^9$) – denoting that the Feng Shui at coordinates $(x, y)^T$ is $(v_x, v_y) \cdot (x, y)^T = v_x \cdot x + v_y \cdot y$.
- one line containing N ($3 \leq N \leq 10^5$) – the number of corners in the polygonal representation of your apartment
- N lines giving the polygon in either clockwise or counterclockwise order, with the i-th line containing x_i and y_i ($-10^6 \leq x_i, y_i \leq 10^6$) – the coordinates of the i-th polygon vertex.

Output

Output your apartment's Feng Shui. It can be shown that the result is always an integer.

Sample input and output

Input	Output
7 3 3 1 3 3 7 2 3	42

Problem B: Baking Cookies

Advanced Algorithms for Programming Contests

Restrictions

Time: 4 seconds
Memory: 512 MB

Problem description

Christmas is mere months away and, as is tradition, you plan to bake lots of cookies and gift them to your friends. Truth be told, you always bake so many cookies that you need to start in July. However, thinking back at last year's difficulties, you are a bit hesitant to start this time. Instead you have decided to analyze your baking process, to see if you could make any simple improvements.

So far, the baking procedure looks as follows: You bake exactly k batches of cookies, each of which consists of exactly m cookies. Most of the time, not all cookies in a batch end up good enough to be used as gifts. Thus you simply eat those imperfect cookies yourself and only collect the good ones. Finally, after all batches are done, you divide the good cookies evenly among your n friends. Of course this doesn't necessarily work out: the number of good cookies might not be divisible by n, in which case even some of them would go to waste (be eaten by you). From your extensive experience, you already know that the probabilities for certain numbers of cookies in a batch to be good are the same across all batches and you even know the probability distribution underlying those numbers.

For now you are interested in a single metric by which you want to assess whether this process is fine as it is: The probability that no good cookies go to waste.

Input

The input consists of

- one line containing integers n, m and k ($2 \leq n, m \leq 10^4, 2 \leq k \leq 1000$) – the number of friends you have, the amounts of cookies per batch and the number of batches you intend to make, respectively
- one line containing $m + 1$ decimal numbers p_0,...,p_m ($0 \leq p_i \leq 1$, $\sum_{i=0}^{m} p_i = 1$), where p_i is the probability that a batch has exactly i good cookies.

Output

Output the probability for the overall number of good cookies to be divisible by n. Your result should have precision 10^{-4}.

Sample input and output

Input	**Output**
2 4 123 0.1 0.0 0.3 0.0 0.6	1.00000000
73 3 25 0.0 0.01 0.99 0.0	0.00000000
17 4 100 0.0 0.1 0.2 0.3 0.4	0.05880874

Problem C: Teleportation

Advanced Algorithms for Programming Contests

Restrictions

Time: 6 seconds
Memory: 512 MB

Problem description

You recently managed to complete a science project you had been working on for a while: a teleportation device. While the teleportation itself already works like a charm, there's still a pretty significant limitation to the device's practicality: It doesn't enable its users to decide *where* they are teleported. Instead, this appears to be decided somewhat randomly.

Through tedious experiments you have already figured out what you believe to be the exact mechanism underlying the device's choice of destination: Basically, if the device's position before the teleport is interpreted as a point $(x, y, z)^T \in \mathbb{R}^3$, then the changes Δx, Δy and Δz in the individual components are chosen independently according to the respective probability distributions p_x, p_y and p_z, which are only supported on integers ranging from $-n$ to n (for some n you also determined). The teleportation then moves the device and its occupants to the point with coordinates $(x+\Delta x, y+\Delta y, z+\Delta z)^T$.

Having understood all this, you would now like to analyze certain properties of this procedure. First of all, you want to know the key characteristic of any randomized teleportation device: The expected distance between its positions before and after a single teleportation.

Input

The input consists of

- one line containing n ($1 \leq n \leq 1000$) – the maximum distance you can teleport in any axis-direction
- one line containing $2n+1$ decimal numbers $p_{x,-n}, p_{x,-n+1}, ..., p_{x,n-1}, p_{x,n}$ ($0 \leq p_{x,i} \leq 1$, $\sum_{i=-n}^{n} p_{x,i} = 1$), where $p_{x,i}$ is the probability to move from a coordinate with x-component a to one with x-component $a+i$
- one line containing $2n+1$ decimal numbers $p_{y,-n}, p_{y,-n+1}, ..., p_{y,n-1}, p_{y,n}$ ($0 \leq p_{y,i} \leq 1$, $\sum_{i=-n}^{n} p_{y,i} = 1$), where $p_{y,i}$ is the probability to move from a coordinate with y-component a to one with y-component $a+i$

- one line containing $2n+1$ decimal numbers $p_{z,-n}, p_{z,-n+1}, ..., p_{z,n-1}, p_{z,n}$ ($0 \leq p_{z,i} \leq 1$, $\sum_{i=-n}^{n} p_{z,i} = 1$), where $p_{z,i}$ is the probability to move from a coordinate with z-component a to one with z-component $a+i$.

Output

Output the expected distance covered by a single teleportation. Your result should have precision 10^{-4}.

Sample input and output

Input	**Output**
1 0.5 0 0.5 0.4 0 0.6 0.3 0 0.7	1.73205081
1 0.5 0 0.5 0.4 0.2 0.4 0.3 0.4 0.3	1.53363836

Problem D: Deficient Data

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You just stumbled upon a strange device that generates three-dimensional data points. At first you thought they were entirely random, but by now you are sensing a pattern – at least a certain fraction $p \geq 10\%$ of them appear to lie on a plane. Naturally, you now want to analyze the data and find the plane-model that is optimal in that it fits as many of the generated points as possible.

Input

The input consists of

- one line containing n ($5 \leq n \leq 10^4$) – the number of data points
- n lines giving the data points $a_1, ..., a_n$, with the i-th line containing x_i, y_i and z_i ($-10^9 \leq x_i, y_i, z_i \leq 10^9$) – denoting that $a_i = (x_i, y_i, z_i)^T$.

For simplicity, the given points will be pairwise distinct and do not all lie on the same line.

Output

Output two lines, one giving the number m of points described by the optimal plane and the other giving m numbers $i_1, ..., i_m$ – the 0-based indices of these data points in ascending order. If there are several answers, output any.

Sample input and output

Input	Output
4 0 0 0 1 0 0 0 1 0 -1 0 0	4 0 1 2 3

Problem E: Gravel Plant

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Your local gravel plant is in trouble: Its new manager has tried to force the workers to make major changes to how the plant is run in order to increase its efficiency. Rather than doing so, the workers have decided to go on strike. At his wit's end, the manager consulted you to mediate the conflict. Specifically, he wants you to make some calculations that show who is in the right.

The situation is as follows: Every day the plant receives a fixed amount of stone from a nearby quarry that can then be used to produce a variety of different types of gravel. Naturally, these different end products vary in both their value on the market and the time it takes to make them. The plant is pretty free in choosing what amount of each type of gravel it produces, except that the daily delivery has to be used up entirely (the current contract doesn't allow for changes to the delivered amount and there is not enough space on the plant to store unused material over extended periods of time). With this restriction there comes another one: While, at first glance, it might seem like the best idea would be to just produce the most profitable type of gravel, this would simply take too long – there would still be leftover raw material when the next load arrives. Instead the plant was run on some decades-old schedule its founder conceived of. It seemed to work pretty well, but the manager took issue with the fact that it didn't use the plant's grinders around the clock – for some part of the day they just stood still!

He is convinced that this cannot possibly be optimal, so he asked you to compute two figures: firstly, the maximum daily profit under just the restriction that the delivered stone needs to be used up, and secondly, the maximum daily profit under the additional restriction that every grinder needs to run all day.

Input

The input consists of

- one line containing two integers s and m ($1 \leq s \leq 10^9, 1 \leq m \leq 100$) – how many kilograms of stone the daily delivery consists of and how many grinders there are on the plant
- one line containing n ($2 \leq n \leq 10^5$) – the number of different types of gravel the plant can produce
- one line containing integers $p_1, ..., p_n$ ($1 \leq p_i \leq 10^5$), where p_i is the profit gained per kilogram of type i gravel (selling price minus raw material price minus production costs)
- one line containing integers $t_1, ..., t_n$ ($1 \leq t_i \leq 10^5$), where t_i is the number of seconds it takes a single grinder to grind a kilogram of stone into type i gravel (this rate is independent of the actual amount – accordingly, grinding 0.123 kilograms of stone into type i gravel would only take $0.123 \cdot t_i$ seconds)

It is to be emphasized that, despite all input parameters being inputs, the plant is not restricted to producing integer or even rational amounts of any given type of gravel (but obviously they need to be nonnegative).

Output

As a day consists of 86400 seconds, the grinders can run for at most $86400 \cdot m$ seconds per day, overall.

First output the maximal daily profit under just the restriction that the daily delivery needs to be used up within the given time frame (i.e. without using the plant's grinders for more than $86400 \cdot m$ seconds overall).

Then output the maximal daily profit under the additional restriction that all grinders need to run around the clock (so that the total time used across all grinders is *equal* to $86400 \cdot m$). It is guaranteed that the daily delivery is more than large enough for the grinders to be running at all times (with some choice of produced types of gravel).

Both outputs should have precision 10^{-4}.

Sample input and output

Input	Output
30516 68 2 2 1 149 1037	61032.00000000 59536.14864865