CCO '09 P6 - A Weighty Problem

Time limit: 0.6s **Memory limit:** 128M

Canadian Computing Competition: 2009 Stage 2, Day 2, Problem 3

You like to make purchases using coins, but you have a problem: you have so much change that it is too heavy in your pocket. You have devised a plan to reduce the weight of your change, and you need to write a program to help you execute it.

Here is your plan. The next purchase you make costs C $(1 \le C \le 100\,000)$ cents, and you know how the store will pay back change if you pay extra. You want to select some of your coins that have total value at least C and make the purchase, such that you minimize the weight of the coins you did not spend added to the weight of the coins the store returns to you.

If the store owes you X cents, then it uses the following algorithm to pay you back. The store gives you the largest denomination coin that has value at most X, and repeats this until all X cents have been paid to you. You may assume the store has an unlimited amount of every denomination of coin.

There are D $(1 \le D \le 100)$ denominations of coins. The i'th denomination $(1 \le i \le D)$ has integer value V_i $(1 \le V_i \le 2\,000)$ in cents and real-valued weight W_i $(0 < W_i < 10)$ in grams. You may assume that one of the denominations has value 1 and that no two denominations have the same value.

You have K $(1 \le K \le 100)$ coins. The j'th coin $(1 \le j \le K)$ is of the denomination with index D_j $(1 \le D_j \le D)$.

In 20% of test cases, $K \le 10$.

Input Specification

The first line contains three integers: C, the cost of the purchase in cents; D, the number of denominations of coins; and K, the number of coins you have.

The next D lines contain an integer V_i, the value of the i'th denomination in cents, and a real number given to two decimal places, W_i, the weight of the i'th denomination in grams.

The next K lines contain an integer D_j, the 1-based denomination of the j'th coin you have.

Output Specification

Output the minimum weight achievable rounded to two decimal places, if you can afford the purchase. If you cannot afford the purchase, print `too poor`.

Sample Input

```
3 4 7
1 1.00
5 2.00
20 9.00
10 1.00
2
2
2
2
2
2
2
```

Description of Sample Input

You have seven 5-cent coins, and are making a purchase of 3 cents. The denominations are 1-cent, 5-cents, 10-cents, and 20-cents, with respective weights of 1 gram, 2 grams, 1 gram and 9 grams.

Output for Sample Input

```
11.00
```

Description of Output for Sample Input

There are two optimal solutions. One is to spend three 5-cent coins, so that the store returns 12 cents to you, in the form of one 10-cent coin and two 1-cent coins. The other is to spend four 5-cent coins, so that the store returns 17 cents to you, in the form of one 10-cent coin, one 5-cent coin, and two 1-cent coins.