There are \( n \) people at the railway station, and each one wants to buy a ticket to go to one of \( k \) different destinations. The \( n \) people are in a queue.

There are \( m \) ticket windows from which tickets can be purchased. The \( n \) people will be distributed in the windows such that the order is maintained. In other words, suppose we number the people \( 1 \) to \( n \) from front to back. If person \( i \) and person \( j \) go to the same window and \( i < j \), then person \( i \) should still be ahead of person \( j \) in the window.

Each ticketing window has an offer. If a person in the queue shares the same destination as the person immediately in front of him/her, a 20\% reduction in the ticket price is offered to him/her.

For example, suppose there are 3 people in the queue for a single ticket window, all with the same destination which costs 10 bucks. Then the first person in the queue pays 10 bucks, and the 2nd and 3rd persons get a discount of 20\% on 10 bucks, so they end up paying 8 bucks each instead of 10 bucks.

Try to distribute the \( n \) people across the \( m \) windows such that the total cost \( S \) paid by all \( n \) people is minimized.

**Input Format**

The first line contains 3 integers:

- \( n \) is the number of people
- \( m \) is the number of ticket windows
- \( k \) is the number of destinations separated by a single space (in the same order)

Then \( k \) lines follow. The \( i^{\text{th}} \) line contains an alphanumeric string \( \text{place}_i \) and an integer \( \text{price}_i \):

- \( \text{place}_i \) is the \( i^{\text{th}} \) destination
- \( \text{price}_i \) is the ticket price for \( \text{place}_i \)

Then \( n \) lines follow. The \( i^{\text{th}} \) line contains an alphanumeric string \( \text{destination}_i \) which is the destination of the \( i^{\text{th}} \) person.

**Constraints**

- \( 1 \leq n \leq 500 \)
- \( 1 \leq m \leq 10 \)
- \( 1 \leq k \leq 100 \)
- The \( k \) available destinations have nonempty and distinct names.
- Each person’s destination appears in the list of \( k \) available destinations.
- \( 0 \leq \text{price}_i \leq 100 \)

**Output Format**

Output \( n + 1 \) lines. The first line contains \( S \), the total cost that is to be minimized. In the \( i^{\text{th}} \) following line, print the ticket window which the \( i^{\text{th}} \) person goes to. The windows are indexed \( 1 \) to \( m \). There may be multiple ways to distribute the people among the windows such that the total cost is minimized; any one will be accepted.

The answer \( S \) will be accepted if it is within an error of \( 10^{-3} \) of the true answer.
Sample Input

5 2 3
CALIFORNIA 10
HAWAII 8
NEWYORK 12
NEWYORK
NEWYORK
CALIFORNIA
NEWYORK
HAWAII

Sample Output

49.2
1
1
2
1
1

Explanation

At the beginning, all the people are in the same queue, and will go to the ticket windows one by one in the initial order.

- \( \{1, 2, 4, 5\} \) will buy ticket in the first window.
- \( \{3\} \) will buy ticket in the second window.

In the first ticket window, \#1 will pay 12 bucks to go to NEWYORK, and \#2 and \#4 have the same destination with the person in front of them, so they will get 20% off, and will pay 9.6 bucks each. \#5 has a different destination, so it will cost him 8 bucks to go to HAWAII.

In the second ticket window, \#3 will pay 10 bucks to go to CALIFORNIA.