Team Mathematics Olympiad is an interesting competition in which a team of n people will answer m questions. Here is a typical (but simplified) problemset:

Question 1. What is the sum of 123 and 987?
Question 2. Let x be the answer of the previous question, how many digit does x have (in decimal form)?
Question 3. What is the smallest 6-digit prime number?
Question 4. How many positive factors does 100! have?
Question 5. Let x be the answer of the previous question, what is the last digit of x^x?

You see, some of the questions make use of the previous question's answer. That means, if your team's answer for the previous question was incorrect, then virtually you have no chance answering this question correctly (i.e. the probability of answering correctly is zero).

The good news is that you can take a look at the problemset and make some quick assessment before you start.

After that, you get a matrix P, in which $P_{i,j}$ is the probability that the i-th person answer the j-th question correctly, given the correct answer to the previous question (if there is).

The bad news is that after the quick assessment, the problemset is taken away, and you're asked to answer the questions in order (i.e. answer question 1 first, then question 2, 3, ... You have to answer each question, if you know you have no chance doing correctly). Before answering each question, your team has to choose one person, then he enters a secret room, read the question, submit his answer and leave the room. Shortly after that, you'll be immediately informed whether the answer is correct, and your team prepare for the next question.

Anyone can go for any question, but the work assignment should be balanced. Let C_i be the total number of question that the i-th person actually answered during the competition, then $max(C_i) - min(C_i)$ should not be more than 1.

Unfortunately, the questions are very lengthy (unlike the simplified problemset above), so nobody can remember enough details to work on a question before going to the secret room.

Similarly, when working on a question in the secret room, nobody can remember the previous question. So if he already knew the previous answer was incorrect, he'll have no chance of answering the current question correctly if it is dependent on the answer of previous question.

Find out a strategy to maximize the expected number of correctly answered questions. Note that the strategy doesn't have to be static: you can make different assignment on different situations.

Input

The first line contains the number of test cases T ($1 \le T \le 100$). The first line of each test case contains two integers n and m ($2 \le n \le 5$, $n \le m \le 30$), the number of team members and the number of questions.

The next line begins with an integer k ($0 \le k \le m - 1$), the number of questions that needs the answer to the previous question. Then k different integers follows, the list of questions (questions are numbered from 1 to n).

The integers are sorted in increasing order, and does not contain 1 (the first question doesn't have "previous question"). Each of the next n lines contains m real numbers between 0 and 1, the j-th number in the i-th line is $P_{i,j}$.

Output

For each test case, print the serial of output followed by the maximal expected number of correctly answered questions (if you follow the best strategy), to four digits after the decimal point. Look at the output for sample input for details. You can safely assume that errors not exceeding 10^{-5} will be ignored.

Sample Input

```
2
2 2
1 2
0.8 0.1
0.3 0.9
2 4
2 2 4
0.5 0.1 0.5 0.1
0.5 0.9 0.5 0.9
```

Sample Output

```
Case 1: 1.5200
Case 2: 1.9000
```