1011. World Cup Betting (20)

时间限制

400 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

With the 2010 FIFA World Cup running, football fans the world over were becoming increasingly excited as the best players from the best teams doing battles for the World Cup trophy in South Africa. Similarly, football betting fans were putting their money where their mouths were, by laying all manner of World Cup bets.

Chinese Football Lottery provided a "Triple Winning" game. The rule of winning was simple: first select any three of the games. Then for each selected game, bet on one of the three possible results -- namely W for win, T for tie, and L for lose. There was an odd assigned to each result. The winner's odd would be the product of the three odds times 65%.

For example, 3 games' odds are given as the following:

W T L

1.1 2.5 1.7

1.2 3.0 1.6

4.1 1.2 1.1

To obtain the maximum profit, one must buy W for the 3rd game, T for the 2nd game, and T for the 1st game. If each bet takes 2 yuans, then the maximum profit would be (4.1\*3.0\*2.5\*65%-1)\*2 = 37.98 yuans (accurate up to 2 decimal places).

**Input**

Each input file contains one test case. Each case contains the betting information of 3 games. Each game occupies a line with three distinct odds corresponding to W, T and L.

**Output**

For each test case, print in one line the best bet of each game, and the maximum profit accurate up to 2 decimal places. The characters and the number must be separated by one space.

**Sample Input**

1.1 2.5 1.7

1.2 3.0 1.6

4.1 1.2 1.1

**Sample Output**

T T W 37.98

1012. The Best Rank (25)

时间限制

400 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

To evaluate the performance of our first year CS majored students, we consider their grades of three courses only: C - C Programming Language, M - Mathematics (Calculus or Linear Algebra), and E - English. At the mean time, we encourage students by emphasizing on their best ranks -- that is, among the four ranks with respect to the three courses and the average grade, we print the best rank for each student.

For example, The grades of C, M, E and A - Average of 4 students are given as the following:

StudentID C M E A

310101 98 85 88 90

310102 70 95 88 84

310103 82 87 94 88

310104 91 91 91 91

Then the best ranks for all the students are *No.1* since the 1st one has done the best in C Programming Language, while the 2nd one in Mathematics, the 3rd one in English, and the last one in average.

**Input**

Each input file contains one test case. Each case starts with a line containing 2 numbers N and M (<=2000), which are the total number of students, and the number of students who would check their ranks, respectively. Then N lines follow, each contains a student ID which is a string of 6 digits, followed by the three integer grades (in the range of [0, 100]) of that student in the order of C, M and E. Then there are M lines, each containing a student ID.

**Output**

For each of the M students, print in one line the best rank for him/her, and the symbol of the corresponding rank, separated by a space.

The priorities of the ranking methods are ordered as A > C > M > E. Hence if there are two or more ways for a student to obtain the same best rank, output the one with the highest priority.

If a student is not on the grading list, simply output "N/A".

**Sample Input**

5 6

310101 98 85 88

310102 70 95 88

310103 82 87 94

310104 91 91 91

310105 85 90 90

310101

310102

310103

310104

310105

999999

**Sample Output**

1 C

1 M

1 E

1 A

3 A

N/A

1013. Battle Over Cities (25)

时间限制

400 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

It is vitally important to have all the cities connected by highways in a war. If a city is occupied by the enemy, all the highways from/toward that city are closed. We must know immediately if we need to repair any other highways to keep the rest of the cities connected. Given the map of cities which have all the remaining highways marked, you are supposed to tell the number of highways need to be repaired, quickly.

For example, if we have 3 cities and 2 highways connecting city1-city2 and city1-city3. Then if city1 is occupied by the enemy, we must have 1 highway repaired, that is the highway city2-city3.

**Input**

Each input file contains one test case. Each case starts with a line containing 3 numbers N (<1000), M and K, which are the total number of cities, the number of remaining highways, and the number of cities to be checked, respectively. Then M lines follow, each describes a highway by 2 integers, which are the numbers of the cities the highway connects. The cities are numbered from 1 to N. Finally there is a line containing K numbers, which represent the cities we concern.

**Output**

For each of the K cities, output in a line the number of highways need to be repaired if that city is lost.

**Sample Input**

3 2 3

1 2

1 3

1 2 3

**Sample Output**

1

0

0

# 1014. Waiting in Line (30)

时间限制

400 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

Suppose a bank has N windows open for service. There is a yellow line in front of the windows which devides the waiting area into two parts. The rules for the customers to wait in line are:

* The space inside the yellow line in front of each window is enough to contain a line with M customers. Hence when all the N lines are full, all the customers after (and including) the (NM+1)st one will have to wait in a line behind the yellow line.
* Each customer will choose the shortest line to wait in when crossing the yellow line. If there are two or more lines with the same length, the customer will always choose the window with the smallest number.
* Customer[i] will take T[i] minutes to have his/her transaction processed.
* The first N customers are assumed to be served at 8:00am.

Now given the processing time of each customer, you are supposed to tell the exact time at which a customer has his/her business done.

For example, suppose that a bank has 2 windows and each window may have 2 custmers waiting inside the yellow line. There are 5 customers waiting with transactions taking 1, 2, 6, 4 and 3 minutes, respectively. At 08:00 in the morning, customer1 is served at window1 while customer2is served at window2. Customer3 will wait in front of window1 and customer4 will wait in front of window2. Customer5 will wait behind the yellow line.

At 08:01, customer1 is done and customer5 enters the line in front of window1 since that line seems shorter now. Customer2 will leave at 08:02, customer4 at 08:06, customer3 at 08:07, and finally customer5 at 08:10.

**Input**

Each input file contains one test case. Each case starts with a line containing 4 positive integers: N (<=20, number of windows), M (<=10, the maximum capacity of each line inside the yellow line), K (<=1000, number of customers), and Q (<=1000, number of customer queries).

The next line contains K positive integers, which are the processing time of the K customers.

The last line contains Q positive integers, which represent the customers who are asking about the time they can have their transactions done. The customers are numbered from 1 to K.

**Output**

For each of the Q customers, print in one line the time at which his/her transaction is finished, in the format HH:MM where HH is in [08, 17] and MM is in [00, 59]. Note that since the bank is closed everyday after 17:00, for those customers who cannot be served before 17:00, you must output "Sorry" instead.

**Sample Input**

2 2 7 5

1 2 6 4 3 534 2

3 4 5 6 7

**Sample Output**

08:07

08:06

08:10

17:00

Sorry

1014. Waiting in Line (30)

时间限制

400 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

Suppose a bank has N windows open for service. There is a yellow line in front of the windows which devides the waiting area into two parts. The rules for the customers to wait in line are:

* The space inside the yellow line in front of each window is enough to contain a line with M customers. Hence when all the N lines are full, all the customers after (and including) the (NM+1)st one will have to wait in a line behind the yellow line.
* Each customer will choose the shortest line to wait in when crossing the yellow line. If there are two or more lines with the same length, the customer will always choose the window with the smallest number.
* Customer[i] will take T[i] minutes to have his/her transaction processed.
* The first N customers are assumed to be served at 8:00am.

Now given the processing time of each customer, you are supposed to tell the exact time at which a customer has his/her business done.

For example, suppose that a bank has 2 windows and each window may have 2 custmers waiting inside the yellow line. There are 5 customers waiting with transactions taking 1, 2, 6, 4 and 3 minutes, respectively. At 08:00 in the morning, customer1 is served at window1 while customer2is served at window2. Customer3 will wait in front of window1 and customer4 will wait in front of window2. Customer5 will wait behind the yellow line.

At 08:01, customer1 is done and customer5 enters the line in front of window1 since that line seems shorter now. Customer2 will leave at 08:02, customer4 at 08:06, customer3 at 08:07, and finally customer5 at 08:10.

**Input**

Each input file contains one test case. Each case starts with a line containing 4 positive integers: N (<=20, number of windows), M (<=10, the maximum capacity of each line inside the yellow line), K (<=1000, number of customers), and Q (<=1000, number of customer queries).

The next line contains K positive integers, which are the processing time of the K customers.

The last line contains Q positive integers, which represent the customers who are asking about the time they can have their transactions done. The customers are numbered from 1 to K.

**Output**

For each of the Q customers, print in one line the time at which his/her transaction is finished, in the format HH:MM where HH is in [08, 17] and MM is in [00, 59]. Note that since the bank is closed everyday after 17:00, for those customers who cannot be served before 17:00, you must output "Sorry" instead.

**Sample Input**

2 2 7 5

1 2 6 4 3 534 2

3 4 5 6 7

**Sample Output**

08:07

08:06

08:10

17:00

Sorry