4th BUTE International
24-hour Programming Contest

Qualifying Round
Problem Set

March 20, 2004

http://www.challenge24.org

FORNAX

The main sponsor of the event is is Fornax Co.

Microsoft®

International Platinum Sponsor

Lufthansa Systems

Gold Grade Sponsor

NOKIA

CONNECTING PEOPLE

Interware

Red Bull®

Silver Grade Sponsors

John von Neumann Computer Society, IEEE

Professional Sponsors

General Notes

Please read this page carefully!

- There are six problems A, B, C, D, E, and F.
- To solve the problems, you can use any platform, operating system, programming language, development tools, libraries, etc. but you must not use any external human resources.
- The input file is always a text file. The lines are separated by a single line feed character (character code 10), the last line of the file is also terminated by an LF character. The output of your programs also has to follow this convention. Moreover, there cannot be trailing space characters at the end of the lines.
- There are 10 test cases for each problem. If not written otherwise, then the input files for problem X are called `X-1.in`, `X-2.in`, . . . , where X is one of A, B, C, D, E, and F. The output files produced by your program should be called `X-1.out`, `X-2.out`, etc.
- We check your solution by comparing your submitted outputs to our reference outputs. We accept an output file for a test case if it is *exactly* the same as ours. We do a simple binary comparison and we do not tolerate even a single bit of difference. Therefore please follow exactly our specifications with respect to spaces, new line characters, upper case/lower case, etc.
- We have provided some sample input/output files for the problems. The first sample input file for Problem X is called `sample-X-1.in`, a correct solution for this test case is given in the file `sample-X-1.out`. We suggest you to check your program with the sample files first, and submit it only if it your output matches the sample outputs.
- To submit your solution, you should login at `http://www.challenge24.org/reg.php?cmd=login` using your team name and password. Then follow the *Electronic contest* link.
- You are required to submit exactly *one .zip file* called `X.zip` for each problem X. This .zip file should contain the output files for each test case of the given problem. Please note: using .zip compression is a must (we *do not* accept .tgz files)! Multiple uploads per problem is allowed, however, only the last .zip file is judged. The output files should be in the root directory of the .zip file, do not make any subdirectories.
- If your output is correct for a test case, then you will get 1 point. You get the point even if your solution for the other test cases are incorrect. Note that we check only your *last* submission for the problem. Therefore if you submit a correct solution, and later you submit an incorrect solution for the test case, then you will get no points for this test case.
- The 20 teams scoring the highest points will gain the right to participate in the final at Budapest, Hungary April 30-May 2, 2004. In case of a tie, the sum of the submission times decides the order. *Submission time* of a problem is the time (in minutes) elapsed between the beginning of the contest (10 am) and the last upload for this problem; or 0 if you received 0 point for the given problem. We always use server time.
- If you find a problem statement ambiguous, questions should be sent directly to the e-mail address `challenge24@challenge24.org`. You must not use `contest2004@challenge24.org`! Questions are answered privately.
- In the first hour of the contest, we also send replies immediately to `contest2004@challenge24.org`. Later a digest of the answers is sent in every 60 minutes. Please check the mailing list regularly so you don't miss any important announcement.

Problem A: Error Correction

Introduction

Captain Jean-Pierre Spock received an extremely important transmission via a sub-ether quantum hyperlink. Unfortunately, the sub-ether quantum hyperlink communication software is still under beta testing and some of the characters in the message were received incorrectly. Your task is to decode the garbled message. We know the language of the message, and we have a dictionary of all the possible words that could appear in the original text. You have to decode the message by examining each word in the text, and determining which word it could have been originally.

Input

Each test case consists of two input files: the garbled message and the dictionary (the text file containing the message has extension `'.in'`, while the extension of the dictionary file is `'.dict'`). In the message some of the characters are replaced by the character `'*'`, meaning that the character was received incorrectly (the character `'*'` does not appear in the original message). For some weird reason, the errors appeared only when the letters `'a'-'z'`, `'A'-'Z'` were transmitted, the other characters were always received correctly.

The dictionary contains all the words that appear in the original message (it can contain words that do not appear in the message). A word is defined as a sequence of characters `'a'-'z'` or `'A'-'Z'` such that the character before and after the word is not a letter. Only the lower case version of the words appear in the dictionary (using only the letters `'a'-'z'`), but in the message any (or all) character of word can be in upper case. For example, if the word `'holodeck'` is present in the dictionary, then the message may contain `'holodeck'`, `'Holodeck'`, `'HoloDeck'`, or `'HOLODECK'`. The words in the dictionary are separated by a new line character 10. (The last word is also followed by a new line character.)

Output

For each test case, you have to output the original version of the message. For each word in the input that contains the character `'*'`, you have to guess what the word in the original message was, and replace the missing characters with the correct letters. Always use lower case letters when replacing a `'*'` (but leave the other characters of the word as they appear in the message). It can be assumed that the problem is unambiguous: for each word in the message, there is only one correct possibility in the dictionary. Other than replacing the characters `'*'` with the correct letters, do not modify the text in any way. In particular, the end of line characters should appear in the output exactly as in the input.

Sample Input

Message:

```
one,O*E,f**r (th*ee t*o) *ix
```

Dictionary:

```
one
two
three
four
five
six
```

Sample Output

Original message:

```
one,OnE,four (three two) six
```

Problem B: Knights of the Round Table

Introduction

King Arthur has summoned the valiant knights of his kingdom to Camelot. In the Castle of Camelot, the knights sit around a large round table and discuss how to defend the country against its enemies. However, some of the knights do not trust each other, so the order of the knights around the table has to be determined in such a way that ensures no fighting will start during the council. Moreover, some of the knights (for example, the brave Sir Robin) are so afraid of the other knights that they will only sit at the table if a trusted ally sits beside them. Your task is to list all the possible seating arrangements of the knights that satisfy the requirements.

Input

Each input file begins with two numbers: n ($2 \leq n \leq 26$), the number of knights, and m ($1 \leq m \leq 100$), the number of requirements that has to be satisfied. The knights are codenamed `'a'`, `'b'`, ..., `'z'`.

The first line of the input is followed by m lines, one line for each requirement. The requirements can be of two types. The requirement

```
b hates a and z and t
```

means that knight `'b'` cannot sit beside either `'a'`, `'z'`, or `'t'`, otherwise they will kill each other. The line

```
r needs q or t or m
```

means that kinght `'r'` sits to the table only if at least one of `'q'`, `'t'`, or `'m'` sits besides him.

Output

For each input file, you have to output all the possible seating arrangements for the knights. Each line of the output should be n characters long (terminated by a new line character 10). These n characters describe the positions of the knights around the table. The first character is the name of the knight sitting at Seat 1, the second character is the name of the knight sitting at Seat 2, etc. Note that seating assignments that differ only by a rotation or by reversing the order count as different solutions: `abdce`, `bdcea` and `ecdba` all appear in the output below. The lines in the output should be sorted alphabetically. It can be assumed that there is always at least one possible solution to the problem. Do not forget that the first knight is the neighbour of the last knight!

Sample Input

```
5 2
a hates c
b needs d
```

Sample Output

```
abdce
adbce
aecbd
aecdb
baecd
bcead
bdaec
bdcea
cbdae
cdbae
ceabd
ceadb
daecb
dbaec
dbcea
dceab
eabdc
eadbc
ecbda
ecdba
```

Problem C: TV programming

Introduction

BanduTV is the most popular TV channel in Bandulu. The channel has a fixed schedule: every day the same types of programs are shown during the same time periods. For example, it is possible that every day there is news from 18:00 to 19:00, a soap opera from 19:00 to 20:00, a movie from 20:00 to 22:00, music from 22:00 to 02:00, etc. Every year at the start of the season, the program director of the channel decides the schedule of the programs. The different departments of the channel propose several possible programs. Each proposal contains a time interval for the program, and the profit expected from it. The director chooses the programs in such a way that maximizes the profit. For example, if the following programs are proposed:

1.	16:00–20:00	Sport	1200
2.	18:00–19:00	News	2550
3.	19:00–20:00	Series	1800
4.	20:00–22:00	Movie	2000
5.	22:00–08:00	Music	800
6.	22:00–01:00	Politics	1300
7.	07:00–16:00	Children's Program	4000

then the best thing to do is to is to select 2, 3, 4, 6 and 7 for a total profit of 11650. Notice that a program can extend past midnight, and it is possible that no program is scheduled for certain time intervals. Of course, we cannot select two programs that overlap.

Input

Each input file begins with a number n ($1 \leq n \leq 1000$), the number of program proposals. The first line of the input is followed by n lines, one line for each proposal. Each line contains the start and end time of the program, and the expected profit. The times are given in 24-hour 4-digit form such as `'08:35'`, and are separated by spaces.

Output

Output is a single number, the maximum profit that can be achieved. The number in the output should be terminated by a new line character 10.

Sample Input

```
7
16:00 20:00 1200
18:00 19:00 2550
19:00 20:00 1800
20:00 22:00 2000
22:00 08:00 800
22:00 01:00 1300
07:00 16:00 4000
```

Sample Output

```
11650
```

Problem D: Wizards

Introduction

30th February is the Annual Magician's Day, since it is the birthday of the Great Archmage Luxiputrius. The National Association of Wizards and Sorcerers has decided to organize several shows on this day throughout the country. There are several cities where these events can be held. However, in a given city only one wizard can perform: it would be too dangerous to allow two wizards to cast spells in the same city at the same time. The wizards live in ivory towers, usually far from the cities. They do not like travelling: a wizard refuses to go a distance of more than 50 km from their home (most probably because the popular Teleport Amulets only have a range of 50 km). The National Association of Wizards and Sorcerers would like to organize as many shows as possible on the Annual Magician's Day. Your job is to determine how many shows can be held.

Input

Each input file begins with two numbers n ($1 \leq n \leq 1000$), the number of cities, and m ($1 \leq m \leq 1000$), the number of wizards. This is followed by $n + m$ lines, each containing two coordinates (in km). The first n of these lines describe the position of the cities, the next m lines describe the position of the towers of the wizards. The coordinates are not necessarily integer.

Output

Output is a single number, the maximum number of shows that can be held on one night. At most one show can be organized in every city, and a wizard can perform no more than one show a day. The number in the output should be terminated by a new line character 10.

Sample Input

```
4 4
10 0
70 0
150 0
220 0
60 0
110 0
190 0
0 0
```

Sample Output

```
4
```

Problem E: Numbers

Introduction

Two players A and B play the following game. They start with eight integers x_1, x_2, …, x_8, and they have eight magic numbers c_1, c_2, …, c_8. The following formula is used to calculate the numbers x_i for $i > 8$:

$$x_i = x_{i-1} \cdot c_1 + x_{i-2} \cdot c_2 + x_{i-3} \cdot c_3 + x_{i-4} \cdot c_4 + x_{i-5} \cdot c_5 + x_{i-6} \cdot c_6 + x_{i-7} \cdot c_7 + x_{i-8} \cdot c_8$$

All the calculations are done modulo 1000, so x_i is always a number between 0 and 999. Each player has a goal, which is a sequence of eight numbers. If these numbers appear in the sequence x_i, then the player wins. For example, if the goal of Player A is 500 12 0 8 67 289 901 415 and

$$x_{89} = 11,\ x_{90} = 12,\ x_{91} = 500,\ x_{92} = 12,\ x_{93} = 0,\ x_{94} = 8,\ x_{95} = 67,\ x_{96} = 289,\ x_{97} = 901,\ x_{98} = 415,\ \ldots$$

then Player A wins at x_{98}. To win the game the numbers have to appear exactly in the given order, and there cannot be additional numbers between the eight numbers in the sequence.

Because life is short, and calculating this sequence is a very boring thing to do, your have to write a program that determines who will be the winner.

Input

Each input file consists of four lines, with eight non-negative integers in each line. The first line contains the eight numbers x_1, x_2, …, x_8. The second line contains the magic numbers c_1, …, c_8. The third line contains the goal of A, while the fourth line contains the goal of B.

Output

You have to determine who will win the game and when. If player A wins at x_{98} (that is, his sequence appears on x_{91}, x_{92}, x_{93}, x_{94}, x_{95}, x_{96}, x_{97}, x_{98}) then the output should be the line

```
A wins at 98.
```

The line should be terminated by a new line character 10. It can be assumed that the game will be finished in at most 10000000 steps.

Sample Input

```
1 2 3 4 5 6 7 8
2 0 1 500 101 222 333 444
1 1 1 1 1 1 1 1
438 393 722 198 794 331 878 976
```

Sample Output

```
B wins at 2829.
```

Problem F: Movie

Introduction

Director Stephen Bergspiel would like to shoot a low-budget movie. The film has only two main characters, so only two actors have to be hired. Bergspiel wrote a list of all the actors who would be able to play the roles in the film. He wants to select the two cheapest actors from this list, but to add interest to the film, he wants to select two actors who have never ever appeared together in any film. You have to write a program that finds the two cheapest actors satisfying these requirements.

Input

Each input file begins with a line containing two integers: n ($1 \leq n \leq 100$), the number of suitable actors, and m ($1 \leq m \leq 10000$), the number of films where these actors appeared. The next n lines describe the cost of the n actors, there is a single integer in each line. The next m lines after that describe the m films. Each line begins with an integer k ($1 \leq k \leq 20$), the number of actors in the film. This number is followed by k numbers (separated by spaces). These numbers identify the actors playing in the film, each number is between 1 and n.

Output

You have to pick two actors such that the sum of their cost is minimal, and they never appeared in the same film. You have to output a number, the cost of the best solution. If there is no solution, then output `'No solution.'`. The output should be terminated by a new line character 10.

Sample Input

```
4 4
10000
20000
30000
40000
2 1 3
1 2
3 1 2 4
2 2 3
```

Sample Output

```
70000
```