MIT Programming Contest
Individual Round Problems
2000

1 Fastest Fingers

A television game show host begins each new game by selecting a player as follows. The host asks candidate players to order four items. The first candidate to order the items correctly wins. If there is a tie for the fastest time, or if no one correctly answers, ths host poses a new question. The producers air only questions that select a player. They are unhappy with their current player selection software and are seeking a replacement.

The candidates have at most 30 seconds to answer the question. During this time, they press buttons A, B, C, or D, indicating how to order the items. Pressing a special rub out button marked X erases the last selection (this has no effect if there are no characters to rub out). For example, after pressing BXXACXDXBDC, the candidate has selected the answer ABDC. Each candidate's selection is sent to the software along with a timestamp (from 0 to 300, in tenths of a second) and a number identifying the candidate. A candidate cannot make simultaneous selections (i.e. with the same timestamp), but two different candidates might make selections at the same time. Although the software receives the messages in time sequence for a particular candidate, messages from two different candidates may arrive out of time sequence. For example, the software might receive the following sequence:

Candidate	Timestamp	Selection	Interpretation
1	20	B	Candidate 1 selected B at time 20 (answer: B)
2	10	B	Candidate 2 selected B at time 10 (answer: B)
1	50	C	Candidate 1 selected C at time 50 (answer: BC)
2	40	D	Candidate 2 selected D at time 40 (answer: BD)
1	70	X	Candidate 1 erased C at time 70 (answer:B)
1	110	D	Candidate 1 selected D at time 110 (answer:BD)
1	120	A	Candidate 1 selected A at time 120 (answer:BDA)
2	100	A	Candidate 2 selected A at time 100 (answer:BDA)
2	150	C	Candidate 2 selected C at time 150 (answer:BDAC)
1	170	C	Candidate 1 selected C at time 170 (answer:BDAC)

Suppose that the correct item order is BDAC. To be considered correct, a candidate's final answer must exactly match the correct item order. In this example, both candidates have the correct answer, but Candidate 2 has the faster time (15.0 seconds), and is the player for the next round.

1.1 Input

The input contains several player selection rounds. Each round begins with a line containing two integers m and n separated by whitespace, where $2 \leq m \leq 10$ is the number of candidates and n is the number of messages. The next line contains only the letters A, B, C, and D, in the correct order for that round; each letter appears exactly once and in upper case. The next n lines contain the messages in the following format

candidate timestamp selection

where *candidate* is an integer between 1 and m inclusive that identifies the candidate sending the message, *timestamp* is an integer between 0 and 300 inclusive representing the time in tenths of a second, and *selection* is either A, B, C, D, or X (in upper case) as explained above. Your program must stop processing input when it encounters a data set in which n is 0.

1.2 Output

Begin the output of each player selection round by summarizing the results. List the candidates in order of candidate number and state whether the candidate was correct or incorrect. If the candidate was correct, indicate the time at which the candidate completed the data entry. If a player is selected, indicate which player. If no player is selected, indicate that this question should not be aired and a new question is needed. Leave a blank line between the output for different player selection rounds. Follow the format shown in the sample output.

1.3 Example

Sample Input

```
2 10
BDAC
1 20 B
2 10 B
1 50 C
2 40 D
1 70 X
1 110 D
1 120 A
2 100 A
2 150 C
1 170 C
2 8
ABCD
1 20 A
1 25 B
1 78 D
2 15 C
2 59 D
2 105 A
2 189 B
1 187 C
2 0
```

Sample Output

```
Round #1: 2 candidates
Candidate 1: Correct in 17.0 seconds
Candidate 2: Correct in 15.0 seconds
Candidate 2 is selected as the player for this round.

Round #2: 2 candidates
Candidate 1: Incorrect
Candidate 2: Incorrect
Don't air this one...we need a new question.
```

2 Unix ls

The computer company you work for is introducing a brand new computer line and is developing a new Unix-like operating system to be introduced along with the new computer. Your assignment is to write the formatter for the `ls` function.

Your program will eventually read input from a pipe (although for now your program will read from the input file). Input to your program will consist of a list of F filenames that you will sort (ascending based on the ASCII character values) and format into C columns based on the length L of the longest filename. Filenames will be between 1 and 60 (inclusive) characters in length and will be formatted into left-justified columns. The rightmost column will be the width of the longest filename and all other columns will be the width of the longest filename plus 2. There will be as many columns as will fit in 60 characters. Your program should use as few rows R as possible with rows being filled to capacity from left to right.

2.1 Input

The input file will contain an indefinite number of lists of filenames. Each list will begin with a line containing a single integer $1 \leq N \leq 100$. There will then be N lines each containing one left-justified filename and the entire line's contents (between 1 and 60 characters) are considered to be part of the filename. Allowable characters are alphanumeric (`a` to `z`, `A` to `Z`, and `0` to `9`) and from the following set { `._-` } (not including the curly braces). There will be no illegal characters in any of the filenames and no line will be completely empty.

Immediately following the last filename will be the N for the next set or the end of file. You should read and format all sets in the input file.

2.2 Output

For each set of filenames you should print a line of exactly 60 dashes (`-`) followed by the formatted columns of filenames. The sorted filenames 1 to R will be listed down column 1; filenames $R+1$ to $2R$ listed down column 2; etc.

2.3 Sample Input

```
10
long-file-name
the longest file name
short
tiny
abc
mid-length
88888888.333
longer_filename
2short4me
another long name
12
Weaser
Alfalfa
Stimey
Buckwheat
Porky
Joe
Darla
Cotton
Butch
Froggy
Mrs_Crabapple
P.D.
19
```

```
Mr._French
Jody
Buffy
Sissy
Keith
Danny
Lori
Chris
Shirley
Marsha
Jan
Cindy
Carol
Mike
Greg
Peter
Bobby
Alice
Ruben
```

2.4 Sample Output

```
------------------------------------------------------------
12345678.123         size-1
2short4me            size2
mid_size_name        tiny
much_longer_name     very_long_file_name
shorter              size3
------------------------------------------------------------
Alfalfa        Cotton         Joe            Stimey
Buckwheat      Darla          Mrs_Crabapple  Weaser
Butch          Froggy         Porky          P.D.
------------------------------------------------------------
Alice       Chris       Jan         Marsha      Shirley
Bobby       Cindy       Jody        Mike        Sissy
Buffy       Danny       Keith       Mr._French  Ruben
Carol       Greg        Lori        Peter
```

3 Scheduling Lectures

You are teaching a course and must cover n ($1 \leq n \leq 1000$) topics. The length of each lecture is L ($1 \leq L \leq 500$) minutes. The topics require $t_1, t_2, \ldots, t_n$ ($1 \leq t \leq L$) minutes each. For each topic, you must decide in which lecture it should be covered. There are two scheduling restrictions:

1. Each topic must be covered in a single lecture. It cannot be divided between two lectures. This reduces discontinuity between lectures.
2. Topic i must be covered before topic $i+1$ for all $1 \leq i \leq n$. Otherwise, students may not have the prerequisites to understand topic $i+1$.

With the above restriction, it is sometimes necessary to have free time at the end of a lecture. If the amount of free time is at most 10 minutes, the students will be happy to leave early. However, if the amount of free time is more, they would feel that their tuition fees are wasted. Therefore, we will model the dissatisfaction index (DI) of a lecture by the formula:

$$DI = \begin{cases} 0 & \text{if } t = 0, \\ -C & \text{if } 1 \leq t \leq 10, \\ (t-10)^2 & \text{otherwise,} \end{cases}$$

where C is a positive integer, and t is the amount of free time at the end of a lecture. The total dissatisfaction index is the sum of the DI for each lecture.

For this problem, you must find the minimum number of lectures that is needed to satisfy the above constraints. If there are multiple lecture schedules with the minimum number of lectures, also minimize the total dissatisfaction index.

3.1 Input

The input consists of a number of cases. The first line of each case contains the integer n, or 0 if there are no more cases. The next line contains the integers L and C. These are followed by n integers $t_1, t_2, \ldots, t_n$.

3.2 Output

For each case, print the case number, the minimum number of lectures used, and the total dissatisfaction index for the corresponding lecture schedule on three separate lines. Output a blank line between cases.

3.3 Sample Input

```
6
30 15
10
10
10
10
10
10
10
120 10
80
80
10
50
30
20
40
30
120
120
0
```

3.4 Sample Output

```
Case 1:

Minimum number of lectures: 2
Total dissatisfaction index: 0

Case 2:

Minimum number of lectures: 6
Total dissatisfaction index: 2700
```

4 Word Amalgamation

In millions of newspapers across the United States there is a word game called Jumble. The object of this game is to solve a riddle, but in order to find the letters that appear in the answer it is necessary to unscramble four words. Your task is to write a program that can unscramble words.

4.1 Input

The input file contains four parts: 1) a dictionary, which consists of at least one and at most 100 words, one per line; 2) a line containing `XXXXXX`, which signals the end of the dictionary; 3) one or more scrambled 'words' that you must unscramble, each on a line by itself; and 4) another line containing `XXXXXX`, which signals the end of the file. All words, including both dictionary words and scrambled words, consist only of lowercase English letters and will be at least one and at most six characters long. (Note that the sentinel `XXXXXX` contains uppercase `X`'s.) The dictionary is not necessarily in sorted order, but each word in the dictionary is unique.

4.2 Output

For each scrambled word in the input, output an alphabetical list of all dictionary words that can be formed by rearranging the letters in the scrambled word. Each word in this list must appear on a line by itself. If the list is empty (because no dictionary words can be formed), output the line "`NOT A VALID WORD`" instead. In either case, output a line containing six asterisks to signal the end of the list.

4.3 Example

Sample Input

```
tarp
given
score
refund
only
trap
work
earn
course
pepper
part
XXXXXX
resco
nfudre
aptr
sett
oresuc
XXXXXX
```

Sample Output

```
score
******
refund
******
part
tarp
trap
******
NOT A VALID WORD
******
course
******
```

This page intentionally left blank.

5 Swiss Draw

Many sports and games hold tournaments to determine at least a winner and, very often, a ranking or ordering as well. In two player games (such as Tennis, Chess and Scrabble), the two most common forms of tournament are "knockout" (usually based on an initial ranking or "seeding") and "round robin" (where everybody plays everybody else). The disadvantage in knockout is that a promising newcomer could meet a very much stronger player early in the tournament and not reach his true position. Round Robin eliminates this but at a hugh cost in time—a Round Robin involving 128 players needs 127 rounds whereas it would take only 7 rounds in a knockout competition.

An alternative know as Swiss Draw is very popular in games such as Scrabble. To maximize competition, any one player will play any other player no more than once. After each round, players are ranked on the number of games they have won, where a draw is equal to half a win (more is better) and, within that, by "spread"—the cumulative difference between their scores and their opponents' scores (again bigger is better). If by chance two or more players tie in this ranking, then they appear in inverse order of their previous ranking, i.e. the initially lower-ranked players move ahead. In each round each player either plays someone above them or the highest ranked player below them that allows everyone to play someone they have not played before. The input will specify the (usually random) ordering before the first game.

Write a program to determine the final ranking of a group of Scrabble players, given the initial draw and the scores for each individual for each round.

5.1 Input

Input will consist of one or more scenarios. The first line of each scenario will consist of two integers, P and R, ($16 \leq P \leq 64$, $4 \leq R \leq P/4$) denoting the nubmer of players (a multiple of two) and the number of rounds respectively. This will be followed by P lines, each line consisting of a name (a string of 1 through 20 alphabetic characters without any spaces) followed by R integers (separated from each other and the name by at least one space) representing the R scores for that individual. The list will be in the initial order of play, thus in the first round player $2n+1$ played player $2n+2$ ($0 \leq n < P/2$). Input will be terminated by a line containing two zeroes (i.e. P and R both zero).

5.2 Output

Output will consist of a list of all the players ranked according to the above criteria, together with the number of wins and the spread. Note that a draw is counted as half a win, so indicate an odd number of draws by a plus sign (+) after the number of wins. The name is left justified in a field of width 20, the number of wins is right justified in a field of width 3, specification of draws occupies 1 character position and the spread is right justified in a field of width 6. Leave one blank line between scenarios.

5.3 Example

Sample Input

```
16 4
Absalom  280 334 319 426
Betsheba 374 514 459 417
Carolyne 318 415 445 481
Davidian 402 361 375 278
Eleanor  425 302 447 522
Frances  425 513 306 327
Gabriel  330 337 365 398
Hermione 539 254 442 450
Ishmael  485 305 540 522
Jeremaih 288 295 367 476
Kenneth  532 304 452 445
Laurence 426 437 260 474
Meredith 438 489 274 475
Nicholas 307 357 380 492
Octavia  426 498 305 497
Patricia 333 253 370 412
0 0
```

Sample Output

```
Ishmael              4    619
Meredith             3    247
Carolyne             3    186
Betsheba             3    158
Kenneth              3    126
Eleanor              2+   -11
Hermione             2    264
Laurence             2    -93
Nicholas             2   -114
Davidian             2   -150
Frances              1+  -119
Octavia              1    -85
Absalom              1   -187
Jeremaih             1   -188
Gabriel              1   -338
Patricia             0   -315
```

6 Javelin Throwing

The coaches for the US Olympic Track Team, Pierre and Ada, fear that their star javelin thrower has not been given a "fair go" by the Olympic referees. They have purchased two distance measuring devices (DMD) to help them check on the accuracy of the measurements made by the referees. There are several elimination rounds for which each coach positions him/herself in either coaches area (which include the boundaries), such that the two coaches are on opposite sides of the throwing zone. At the beginning of each round the coaches measure their separation from each other and the distance of each from the launching point. During a round the coaches do not move. For each throw in each round they record their distances from the landing point, and calculate the distance that should have been awarded to their javeling thrower. You may assume that the javeling will always land somewhere in the landing area (which includes its boundary).

Put a figure here!

The following measurements are made, as indicated in the figure above:

x Pierre's distance from the launching point

y Ada's distance from the launching point

z distance between Ada and Pierre

a Pierre's distance from the javelin's landing point

c Ada's distance from the javelin's landing point

Your program is to determine b, the actual distance that the javelin traveled (in meters, to the nearest hundredth of a meter) from the other measurements (given in meters, to the nearest hundredth of a meter).

6.1 Input

There will be several groups of input data, each representing a round. The first line of each group will have an integer and three real numbers, n, x, y, and z separated by white space. The first number n is the number of throws in the current round, such that $0 \leq n \leq 5$. The other distance limits are $10 \leq x \leq 100$, $10 \leq y \leq 100$, and $5 \leq z \leq 100$. The next n lines will have pairs of real numbers, representing measurements a and c, such that $0 \leq a \leq 100$, and $0 \leq c \leq 100$. The input data is terminated by a value of $n = 0$, followed by three zeroes, and this data is not processed.

6.2 Output

The output should be labeled by the number of the round, and followed by the distance of each throw for that round (each on a separate line). Formatting should be as in the sample output.

6.3 Example

Sample Input

```
3 50.00 50.00 75.00
50.00 50.00
60.00 40.00
30.00 75.00
2 30.00 60.00 87.50
55.55 66.66
33.33 88.88
0 0 0 0
```

Sample Output

```
Round 1
1. 66.14
2. 66.30
3. 69.96

Round 2
1. 52.77
2. 48.81
```

This page intentionally left blank.

7 Most Wanted Word

Write a program to find the most frequent word in a file of text. A word is any non-empty continuous sequence of alphabetic characters. Case is not regarded as significant, so the words "bother" and "BOTHeR" should be considered the same word.

Any non-alphabetic characters (including control characters such as newlines) can be used to separated words. Thus "isn't" is counted as two words, the second of which is a one letter word consisting only of the letter 't'.

If more than one word appears with the maximum frequency, then the first word to reach the maximum frequency is required.

Input will consist of a set of paragraphs each terminated by a single line containing only the character '`#`' which will not otherwise occur anywhere in the text. No paragraph will contain more than 2000 different words, and no word will be more than 20 letters long. The input is terminated by a test case with no words at all. This case should produce no output.

For each paragraph, your program should print one line containing the frequency of the most frequent word, right justified in a field of width 4, followed by a space and the most frequently occuring word itself, entirely in lower case.

7.1 Sample Input

```
This is a simple file of test data, which should
not cause your program any trouble. Do note that
it contains several punctuation characters. Of course,
this is not really a problem, because such characters
are treated in the same way as spaces.
#
Don't use contractions; it isn't nice.
#
aBc def AbC def dfe ABC
#
123
#
```

7.2 Sample Output

```
   2 of
   2 t
   3 abc
```

This page intentionally left blank.