湖南省第九届大学生计算机程序设计竞赛

The Ninth Hunan Collegiate Programming Contest

主办：湖南省教育厅

协办：湖南省高等教育学会计算机教育专业委员会

承办：湖南人文科技学院

2013 年 10 月 13 日

本次比赛 11 道题目，共 17 页。如有缺页，请立即通知赛场工作人员。

所有题目均采用标准输入输出，请不要读写任何文件。

所有题目的正确输出均是惟一的。你的输出只有和正确输出完全一致时才能通过。

题目 A

近似回文词

输入一行文本，输出最长近似回文词连续子串。所谓近似回文词是指满足以下条件的字符串：

1. S 以字母开头，字母结尾

2. a(S)和 b(S)最多有 2k 个位置不同，其中 a(S)是 S 删除所有非字母字符并且把所有字母转化成小写之后得到的串，b(S)是 a(S)的逆序串。

比如当 k=1 时，Race cat 是一个近似回文词，因为 a(S)=racecat 和 b(S)=tacecar 只有 2 个位置不同。

输入

输入包含不超过 25 组数据，每组数据包含两行。第一行是整数 k（0<=k<=200），第二行为字符串 S，包含至少一个字母但不超过 1000 个字符（换行符不算）。S 只包含字符、空格和其他可打印字符（比如逗号，句号），并且不会以空白字符开头。

输出

对于每组测试数据，输出最长近似回文子串的长度和起始位置（S 的第一个字符是位置 1）。如果有多个最长近似回文子串解，起始位置应尽量小。

样例输入	样例输出
1 Wow, it is a Race cat! 0 abcdefg 0 Kitty: Madam, I'm adam.	Case 1: 8 3 Case 2: 1 1 Case 3: 15 8

题目 B
一行盒子

你有一行盒子，从左到右依次编号为 1, 2, 3,…, *n*。你可以执行四种指令：

- 1 X Y 表示把盒子 X 移动到盒子 Y 左边（如果 X 已经在 Y 的左边则忽略此指令）。
- 2 X Y 表示把盒子 X 移动到盒子 Y 右边（如果 X 已经在 Y 的右边则忽略此指令）。
- 3 X Y 表示交换盒子 X 和 Y 的位置。
- 4 表示反转整条链。

指令保证合法，即 X 不等于 Y。例如，当 n=6 时在初始状态下执行 1 1 4 后，盒子序列为 2 3 1 4 5 6。接下来执行 2 3 5，盒子序列变成 2 1 4 5 3 6。再执行 3 1 6，得到 2 6 4 5 3 1。最终执行 4，得到 1 3 5 4 6 2。

输入

输入包含不超过 10 组数据，每组数据第一行为盒子个数 n 和指令条数 m（1<=n,m<=100,000），以下 m 行每行包含一条指令。

输出

每组数据输出一行，即所有奇数位置的盒子编号之和。位置从左到右编号为 1~n。

样例输入	样例输出
6 4 1 1 4 2 3 5 3 1 6 4 6 3 1 1 4 2 3 5 3 1 6 100000 1 4	Case 1: 12 Case 2: 9 Case 3: 2500050000

题目 C

字符识别?

你的任务是写一个程序进行字符识别。别担心，你只需要识别 1, 2, 3，如下：

```
.*.  ***  ***
.*.  ..*  ..*
.*.  ***  ***
.*.  *..  ..*
.*.  ***  ***
```

输入

输入仅包含一组数据，由 6 行组成。第一行为字符的个数 n（1<=n<=10）。以下 5 行每行包含 4n 个字符。每个字符恰好占 5 行 3 列，然后是一个空列（用"."填充）。

输出

输出应包含一行，即识别出的各个字符。

样例输入	样例输出
3 .*..***.***. .*....*...*. .*..***.***. .*..*.....*. .*..***.***.	123

Problem D
Damaging Your Spreadsheet
(Spreadsheet Tracking II)

Data in spreadsheets are stored in cells, which are organized in rows (r) and columns (c). Some operations on spreadsheets can be applied to single cells (r,c), while others can be applied to entire rows or columns. Typical cell operations include inserting and deleting rows or columns and exchanging cell contents.

Some spreadsheets allow users to mark collections of rows or columns for deletion, so the entire collection can be deleted at once. Some (unusual) spreadsheets allow users to mark collections of rows or columns for insertions too. Issuing an insertion command results in new rows or columns being inserted before each of the marked rows or columns.

Suppose, for example, the user marks rows 1 and 5 of the spreadsheet on the left for deletion, then marks columns 3, 6, 7, and 9 for deletion, the spreadsheet shrinks to spreadsheet on the right:

	A	B	C	D	E	F	G	H	I
1	**22**	**55**	**66**	**77**	**88**	**99**	**10**	**12**	**14**
2	2	24	**6**	8	22	**12**	**14**	16	**18**
3	18	19	**20**	21	22	**23**	**24**	25	**26**
4	24	25	**26**	67	22	**69**	**70**	71	**77**
5	**68**	**78**	**79**	**80**	**22**	**25**	**28**	**29**	**30**
6	16	12	**11**	10	22	**56**	**57**	58	**59**
7	33	34	**35**	36	22	**38**	**39**	40	**41**

	A	B	C	D	E
1	2	24	8	22	16
2	18	19	21	22	25
3	24	25	67	22	71
4	16	12	10	22	58
5	33	34	36	22	40

If the user marks rows 2, 3 and 5 for insertion, then marks column 3 for insertion, the spreadsheet grows to the one below:

	A	B	C	D	E	F
1	2	24		8	22	16
2						
3	18	19		21	22	25
4						
5	24	25		67	22	71
6	16	12		10	22	58
7						
8	33	34		36	22	40

Now that someone damaged your spreadsheet by issuing several insertion, deletion and exchange operations (details are described below).

Your task is to calculate the number of cells that are kept (not deleted), and the total distance between the original cells and their final locations. If a cell in (x1, y1) was moved to (x2, y2), the total distance is increased by |x1-x2|+|y1-y2|. You also need to determine the final locations of some important data.

Input

The input consists of a sequence of spreadsheets, operations on those spreadsheets, and queries about them. Each spreadsheet definition begins with a pair of integers specifying its initial number of rows (r) and columns (c), followed by an integer specifying the number (n) of spreadsheet operations. Row and column labeling begins with 1. The following n lines specify the desired operations. 1<=r,c<=5000, 1<=n<=50. There will be at least one row and one column at any time.

An operation to exchange the contents of cell (r1, c1) with (r2, c2) is given by: `EX r1 c1 r2 c2.` The four insert and delete commands--DC (delete columns), DR (delete rows), IC (insert columns), and IR (insert rows) are given by: `<command> A` x_1 x_2 `...` x_A, where <command> is one of the four commands; A is a positive integer not greater than 5, and x_1, …, x_A are the labels of the columns or rows to be deleted or inserted before. For each insert and delete command, the order of the rows or columns in the command has no significance. Within a single delete or insert command, labels will be unique.

The operations are followed by an integer Q(Q<=10000), which is the number of queries for the spreadsheet. Each query consists of positive integers r and c, representing the row and column number of a cell in the original spreadsheet. For each query, your program must determine the current location of the data that was originally in cell (r, c).

The end of input is indicated by a row consisting of a pair of zeros for the spreadsheet dimensions.

Output

For each spreadsheet, your program must output its sequence number (starting at 1). In the next line, your program must output the number of cells that are kept, and the total move distance.

For each query, your program must output the original cell location followed by the final location of the data or the word GONE if the contents of the original cell location were destroyed as a result of the operations.

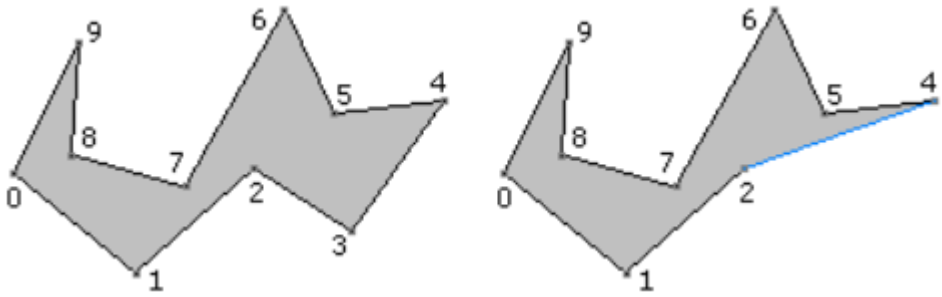
Separate output from different spreadsheets with a blank line.

Sample Input	Output for the Sample Input
7 9 5 DR 2 1 5 DC 4 3 6 7 9 IC 1 3 IR 2 2 4 EX 1 2 6 5 4 4 8 5 5 7 8 6 5 0 0	Spreadsheet #1 There are 25 cell(s) kept, total distance = 29 Cell data in (4,8) moved to (4,6) Cell data in (5,5) GONE Cell data in (7,8) moved to (7,6) Cell data in (6,5) moved to (1,2)

题目 E

割耳法

割耳法可以把一个多边形切成三角形：每次沿着某条对角线切下来一个三角形（称为“耳朵”），n-3 次就能把一个 n 边形切成一个三角形。如下图，三角形{2, 3, 4}被割掉了。



输入一个多边形，怎样做才能让每次切割痕迹的总长度最小？

输入

输入最多包含 30 组测试数据。每组数据第一行为多边形的顶点数 *n*(4<=*n*<=100)。以下 *n* 行描述多边形的各个顶点坐标（均为绝对值不超过 10000 的整数），按照逆时针或者顺时针排列。

输出

对于每组数据，输出切割痕迹总长度的最小值，保留 4 位小数。

样例输入	样例输出
4 0 0 3 0 1 1 0 3 4 0 0 10 0 10 1 0 1	Case 1: 1.4142 Case 2: 10.0499

Problem F
Funny Car Racing

There is a funny car racing in a city with *n* junctions and *m* directed roads.

The funny part is: each road is open and closed periodically. Each road is associate with two integers (*a*, *b*), that means the road will be open for *a* seconds, then closed for *b* seconds, then open for *a* seconds... All these start from the beginning of the race. You must enter a road when it's open, and leave it before it's closed again.

Your goal is to drive from junction *s* and arrive at junction *t* as early as possible. Note that you can wait at a junction even if all its adjacent roads are closed.

Input
There will be at most 30 test cases. The first line of each case contains four integers n, m, s, t (1<=n<=300, 1<=m<=50,000, 1<=s,t<=n). Each of the next m lines contains five integers u, v, a, b, t (1<=u,v<=n, 1<=a,b,t<=10^5), that means there is a road starting from junction u ending with junction v. It's open for a seconds, then closed for b seconds (and so on). The time needed to pass this road, by your car, is t. No road connects the same junction, but a pair of junctions could be connected by more than one road.

Output
For each test case, print the shortest time, in seconds. It's always possible to arrive at t from s.

Sample Input	Output for the Sample Input
3 2 1 3 1 2 5 6 3 2 3 7 7 6 3 2 1 3 1 2 5 6 3 2 3 9 5 6	Case 1: 20 Case 2: 9

题目 G

好老师

我想当一个好老师，所以我决定记住所有学生的名字。可是不久以后我就放弃了，因为学生太多了，根本记不住。但是我不能让我的学生发现这一点，否则会很没面子。所以每次要叫学生的名字时，我会引用离他最近的，我认得的学生。比如有 10 个学生：

A ? ? D ? ? ? H ? ?

想叫每个学生时，具体的叫法是：

位置	叫法
1	A
2	right of A (A 右边的同学)
3	left of D (D 左边的同学)
4	D
5	right of D (D 右边的同学)
6	middle of D and H (D 和 H 正中间的同学)
7	left of H (H 左边的同学)
8	H
9	right of H (H 右边的同学)
10	right of right of H (H 右边的右边的同学)

输入

输入只有一组数据。第一行是学生数 n（1<=n<=100）。第二行是每个学生的名字，按照从左到右的顺序给出，以空格分隔。每个名字要么是不超过 3 个英文字母，要么是问号。至少有一个学生的名字不是问号。下一行是询问的个数 q（1<=q<=100）。每组数据包含一个整数 p（1<=p<=n），即要叫的学生所在的位置（左数第一个是位置 1）。

输出

对于每个询问，输出叫法。注意"middle of X and Y"只有当被叫者有两个最近的已知学生 X 和 Y，并且 X 在 Y 的左边。

样例输入	样例输出
10 A ? ? D ? ? ? H ? ? 4 3 8 6 10	left of D H middle of D and H right of right of H

题目 H

高桥和低桥

有个脑筋急转弯是这样的：有距离很近的一高一低两座桥，两次洪水之后高桥被淹了两次，低桥却只被淹了一次，为什么？答案是：因为低桥太低了，第一次洪水退去之后水位依然在低桥之上，所以不算“淹了两次”。举例说明：

假定高桥和低桥的高度分别是 5 和 2，初始水位为 1

第一次洪水：水位提高到 6（两个桥都被淹），退到 2（高桥不再被淹，但低桥仍然被淹）

第二次洪水：水位提高到 8（高桥又被淹了），退到 3。

没错，文字游戏。关键在于“又”的含义。如果某次洪水退去之后一座桥仍然被淹（即水位不小于桥的高度），那么下次洪水来临水位提高时不能算“又”淹一次。

输入 n 座桥的高度以及第 i 次洪水的涨水水位 a_i 和退水水位 b_i，统计有多少座桥至少被淹了 k 次。初始水位为 1，且每次洪水的涨水水位一定大于上次洪水的退水水位。

输入

输入文件最多包含 25 组测试数据。每组数据第一行为三个整数 n, m, k（$1<=n,m,k<=10^5$）。第二行为 n 个整数 h_i（$2<=h_i<=10^8$），即各个桥的高度。以下 m 行每行包含两个整数 a_i 和 b_i（$1<=b_i<a_i<=10^8$, $a_i>b_{i-1}$）。输入文件不超过 5MB。

输出

对于每组数据，输出至少被淹 k 次的桥的个数。

样例输入	样例输出
2 2 2 2 5 6 2 8 3 5 3 2 2 3 4 5 6 5 3 4 2 5 2	Case 1: 1 Case 2: 3

Problem I
Interesting Calculator

There is an interesting calculator. It has 3 rows of buttons.

Row 1: button 0, 1, 2, 3, ..., 9. Pressing each button *appends* that digit to the end of the display.
Row 2: button +0, +1, +2, +3, ..., +9. Pressing each button *adds* that digit to the display.
Row 3: button *0, *1, *2, *3, ..., *9. Pressing each button *multiplies* that digit to the display.

Note that it never displays leading zeros, so if the current display is 0, pressing 5 makes it 5 instead of 05. If the current display is 12, you can press button 3, +5, *2 to get 256. Similarly, to change the display from 0 to 1, you can press 1 or +1 (but not both!).

Each button has a positive cost, your task is to change the display from x to y with minimum cost. If there are multiple ways to do so, the number of presses should be minimized.

Input
There will be at most 30 test cases. The first line of each test case contains two integers x and y($0<=x<=y<=10^5$). Each of the 3 lines contains 10 positive integers (not greater than 10^5), i.e. the costs of each button.

Output
For each test case, print the minimal cost and the number of presses.

Sample Input	Output for the Sample Input
12 256 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 12 256 100 100 100 1 100 100 100 100 100 100 100 100 100 100 100 1 100 100 100 100 100 100 10 100 100 100 100 100 100 100	Case 1: 2 2 Case 2: 12 3

题目 J

搞笑版费马大定理

费马大定理：当 n>2 时，不定方程 $a^n+b^n=c^n$ 没有正整数解。比如 $a^3+b^3=c^3$ 没有正整数解。为了活跃气氛，我们不妨来个搞笑版：把方程改成 $a^3+b^3=c3$，这样就有解了，比如 a=4, b=9, c=79 时 $4^3+9^3=793$。

输入两个整数 x, y, 求满足 x<=a,b,c<=y 的整数解的个数。

输入

输入最多包含 10 组数据。每组数据包含两个整数 x, y（$1<=x,y<=10^8$）。

输出

对于每组数据，输出解的个数。

样例输入	样例输出
1 10 1 20 123 456789	Case 1: 0 Case 2: 2 Case 3: 16

Problem K
Killer Puzzle

Have you tried this horrible-looking puzzle?

1. Which question is the first question whose answer is b?
(a) 2; (b) 3; (c) 4; (d) 5; (e) 6;

2. The only question that has the same answer as its next question is (e.g. option e means question 6 and 7's answers are the same):
(a) 2; (b) 3; (c) 4; (d) 5; (e) 6;

3. Among the 5 options, which question has the same answer as this question (i.e. question 3)?
(a) 1; (b) 2; (c) 4; (d) 7; (e) 6;

4. How many questions' answer is a?
(a) 0; (b) 1; (c) 2; (d) 3; (e) 4

5. Which of the following questions has the same answer as this question?
(a) 10; (b) 9; (c) 8; (d) 7; (e) 6;

6. The number of questions whose answer is a, equals the number of questions whose answer is:
(a) b; (b) c; (c) d; (d) e; (e) none of above

7. What is the difference of this question's answer and the next question's answer (e.g. the difference of a and b is 1) ?
(a) 4; (b) 3; (c) 2; (d) 1; (e) 0;

8. How many questions' answer is a vowel? (only a and e are vowels. Others are consonants)
(a) 2; (b) 3; (c) 4; (d) 5; (e) 6

9. The number of questions whose answer is a consonant is:
(a) a prime; (b) a factorial; (c) a square number; (d) a cubic number; (e) a multiple of 5

10. The answer of this question is:
(a) a; (b) b; (c) c; (d) d; (e) e;

Note:
1. make sure that your answer is not self-contradicting. For example, the first question's answer can't be b.
2. make sure that for each question, *only* your answer is correct, all other options must be incorrect. For example, if your answer to question 5 is a, then none of question 9, 8, 7, 6's answers can be a!
3. make sure that your answer won't make any question invalid. For example, if question 2 and 3's answer are the same, and question 8,9's answers are also the same, question 2 would be invalid (because no question is "the only question" that satisfying the condition)

It's possible to solve this problem by hand, but as a programmer, solving it with a program is more fun!

How to Solve the Puzzle with a Program

Here's one way: enumerate all possible answers (5^{10} = 9765625), and for each question, check whether only your answer is correct. Pseudo-code:

```
forall(answer_list):
  bad = False
  for testing_question in [1,2,3,4,5,6,7,8,9,10]:
    for testing_option in ["a","b","c","d","e"]:
      # your answer should be correct
      if testing_option == answer_list[testing_question] and
check(testing_question, testing_option) == False:
        bad = True
```

```
      # other options must be incorrect
      if testing_option != answer_list[testing_question] and
check(testing_question, testing_option) == True:
        bad = True
  if not bad:
    print answer_list
```

Here "answer_list" is a list of letters (subscript is 1-based), where the i-th letter is the answer to the i-th question.

Believe or not, the *only* answer is: cdebeedcba (if you prefer to add the question index before each answer, it is 1c2d3e4b5e6e7d8c9b10a)

Amazing, huh? There's more. You wish that your program could solve some other puzzles, but first of all, you need to formulate the puzzle in a formal language.

Formalizing the Puzzle

This problem uses a LISP dialect to represent the puzzle. Don't worry if you don't know LISP, it has a very simple syntax. `(f a b)` means calling a function f with parameters a and b. That's like `f(a, b)` in C/C++/Java. Similarly, `(f a (g b c) d)` is like `f(a, g(b, c), d)` in C/C++/Java.

Here is an example of how to describe a question of the puzzle:

```
3. (equal (answer 3) (answer (option-value)))
a. 1
b. 2
c. 4
d. 7
e. 6
```

There are two very important built-in functions involved:

`(answer idx)`	returns answer_list[idx] in the pseudo-code above.
`(option-value)`	returns the "evaluation result" of testing_option's text, treated as an expression.

In the example above, if testing_option is "c", then (option-value) returns 4 (an integer) because 4 is the expression presented in the option "c" of this question. Note that testing_option's text can be a complex expression instead of a simple value. Refer to Sample Input.

The function `check(testing_question, testing_option)` above can be implemented as follows:

```
check(testing_question, testing_option):
  1. set-up the function (option-value) so that it returns the evaluation
result of testing_option of testing_question
  2. evaluate the lisp expression of testing_question (e.g. the expression
(equal (answer 3) (answer (option-value))) in the example above)
  3. if an unhandled exception is raised during the evaluation, returns False
  4. if the result of step 2 is boolean, return it; otherwise return False
```

There is one special option expression: "none-of-above". The result of "none-of-above" depends on other options' evaluation results. In this problem, there can be at most one "none-of-above" for each question, and it must be the last option.

Details

Here are the details of the LISP dialect used in this problem:

- There are four datatypes: integer, string, boolean and functions.
- There are only two boolean values: true and false. Note that there are no "boolean literal", so you don't care whether to use #t and #f (like in Scheme), or t and nil (like in Common Lisp) to represent boolean constants.
- Integers are always non-negative integers.

- String literals are always enclosed by double quotes, like "a string".
- There is no variable. All the so-called "identifiers" (consisting of letters and hyphens) are always pre-defined functions.

Below is a list of pre-defined functions. Functions starting with ! means it may throw an exception, and functions starting with @ means it can handle exceptions. Like C++/Java/Python, once an exception is raised, the evaluation process is stopped unless a function handles the exception. In the text below, iff means "if and only if".

Basic functions

`(equal a b)`	return true iff a and b are of the same type and are equal. In this problem, you'll never need to compare two functions.
`(option-value)`	discussed above.
`!(answer idx)`	discussed above. If idx is not an integer or is not in 1~n (where n is the number of questions), then raises an exception
`!(answer-value idx)`	Returns the "evaluation result" of option answer_list[idx] of question idx. Also raises an exception on error.

Predicates

Predicate is a special kind of function. It always takes a value of any type and returns a boolean value.

`prime-p`	returns true iff the parameter is a positive prime
`factorial-p`	self-explanatory
`square-p`	self-explanatory
`cubic-p`	self-explanatory
`vowel-p`	returns true iff the parameter is a single letter and is a vowel
`consonant-p`	self-explanatory

Queries and statistics

`!@(first-question pred)`	Return id of the first question that satisfies `pred`. Raises an exception if not found.
`!@(last-question pred)`	Return id of the last question that satisfies `pred`. Raises an exception if not found.
`!@(only-question pred)`	Return id of the only question that satisfies `pred`. Raises an exception if not found or more than one question found.
`@(count-question pred)`	Return the number of questions that satisfies `pred`.
`!(diff-answer idx1 idx2)`	The difference of answers of question idx1 and idx2. Raises an exception on error, otherwise the return value is always 0~m-1, where m is the number of options for each question.

Note that in the first four functions (those with a '@' flag), if an exception was raised when evaluating the predicate, the exception is handled and the predicate is not considered satisfied. For example, if answer_list is "abc", `(count-question (make-answer-diff-next-equal 0))` returns 0 and doesn't raise an exception, even though evaluating the predicate for question 3, i.e. `((make-answer-diff-next-equal 0) 3)` raised an exception. However, all other functions will not handle exceptions. For example, if there are only 3 questions, `(factorial-p (answer-value 5))` will raise an exception instead of returning false.

Predicate generators

There are also functions that can create predicates on-the-fly:

`!(make-answer-diff-next-equal num)`	returns a predicate (p idx) which evaluates (diff-answer idx idx+1) and returns true if the result equals to num. Raises an exception if num is not an integer.
`(make-answer-equal a)`	returns a predicate (p idx) which evaluates (answer idx) and returns true if the result equals a.
`(make-answer-is pred)`	returns a predicate (p idx) which evaluates (answer idx) and returns true if the result satisfies `pred`.
`(make-answer-value-equal a)`	self-explanatory. The predicate evaluates (answer-value idx)
`(make-answer-value-is pred)`	self-explanatory. The predicate evaluates (answer-value idx)
`!(make-is-multiple num)`	returns a predicate (p i) which returns true iff i is an integer and is a multiple of num. Raises an exception if num is not an integer.
`!(make-equal val)`	returns a predicate (p v) which returns true iff (equal v val) is true. Raises an exception if val is neither an integer nor a string.
`(make-not pred)`	returns a predicate (p v) which returns true iff (pred v) is false.

`(make-and pred1 pred2)`	returns a predicate (p v) which returns true iff (pred1 v) and (pred2 v) are both true. Both pred1 and pred2 need to be evaluated. No short-circuit operation should be done.
`(make-or pred1 pred2)`	returns a predicate (p v) which returns true iff at least one of (pred1 v) and (pred2 v) is true. Both pred1 and pred2 need to be evaluated. No short-circuit operation should be done.

For example, `(make-is-multiple 3)` returns a predicate "is a multiple of 3", so `((make-is-multiple 3) 6)` returns true and `((make-is-multiple 3) 10)` returns false. Similarly `(make-not (make-or square-p prime-p))` returns a predicate "neither a square nor a prime".

Input

There will be at most 50 test cases. Each test case begins with two integers n and m(2<=n<=6, 2<=m<=5), the number of questions and the number of options per question. Each question is described with m+1 lines: the question's expression and the options. Questions are numbered 1~n, and options are labeled a~e. Options are valid expressions and will not call option-value (calling option-value makes it recursive!). Each question is followed by a blank line. Most test cases are easy.

Output

For each test case, print the case number in the first line, and a list of answers, one per line, sorted in ascending order. There will always be at least one answer.

Sample Input

```
3 3
(equal (option-value) (count-question (make-
answer-equal "a")))
3
0
1

(equal (option-value) "a")
"c"
"b"
"a"

((option-value) (count-question (make-answer-
equal "c")))
(make-and (make-is-multiple 2) (make-or
factorial-p prime-p))
(make-not prime-p)
"none-of-above"

3 2
(equal (option-value) (answer 2))
"a"
"none-of-above"

(equal (option-value) (first-question (make-
answer-diff-next-equal 0)))
1
2

((option-value) (last-question (make-answer-
equal "b")))
(make-is-multiple 2)
(make-not (make-is-multiple 2))

3 2
(equal (option-value) (answer 1))
"a"
"b"

((option-value) (last-question (make-answer-
```

Output for the Sample Input

```
Case 1:
bcb
cca
Case 2:
aab
Case 3:
aba
Case 4:
ab
ee
Case 5:
ba
```

diff-next-equal 0))) (make-equal 2) "none-of-above" ((option-value) (only-question (make-answer-equal "b"))) (make-is-multiple 2) "none-of-above" 2 5 ((option-value) (diff-answer 1 2)) factorial-p prime-p (make-not square-p) (make-not cubic-p) "none-of-above" (equal (only-question (option-value)) 1) (make-answer-is consonant-p) (make-answer-is vowel-p) (make-answer-value-equal 1) (make-answer-value-is square-p) "none-of-above" 2 2 (option-value) (equal (first-question (make-answer-diff-next-equal 2)) (first-question (make-answer-diff-next-equal 2))) "none-of-above" (equal (option-value) 1) 1 2	