Problem A: Welcome Party

Source file: `welcome.{c, cpp, java}`
Input file: `welcome.in`

For many summers, the Agile Crystal Mining company ran an internship program for students. They greatly valued interns' ability to self-organize into teams. So as a get-to-know-you activity during orientation, they asked the interns to form teams such that all members of a given team either have first names beginning with the same letter, or last names beginning with the same letter. To make it interesting, they asked the interns to do this while forming as few teams as possible.

As an example, one year there were six interns: Stephen Cook, Vinton Cerf, Edmund Clarke, Judea Pearl, Shafi Goldwasser, and Silvio Micali. They were able to self-organize into three teams:

- Stephen Cook, Vinton Cerf, and Edmund Clarke (whose last names all begin with C)
- Shafi Goldwasser and Silvio Micali (whose first names begin with S)
- Judea Pearl (not an interesting group, but everyone's first name in this group starts with J)

As a historical note, the company was eventually shut down due to a rather strange (and illegal) hiring practice---they refused to hire any interns whose last names began with the letter S, T, U, V, W, X, Y, or Z. (First names were not subject to such a whim, which was fortunate for our friend Vinton Cerf.)

Input: Each year's group of interns is considered as a separate trial. A trial begins with a line containing a single integer N, such that $1 \le N \le 300$, designating the number of interns that year. Following that are N lines---one for each intern---with a line having a first and last name separated by one space. Names will not have any punctuation, and both the first name and last name will begin with an uppercase letter. In the case of last names, that letter will have an additional constraint that it be in the range from 'A' to 'R' inclusive. The end of the input is designated by a line containing the value 0. There will be at most 20 trials.

Output: For each trial, output a single integer, k, designating the minimum number of teams that were necessary.

Example Input:	Example Output:
6 Stephen Cook Vinton Cerf Edmund Clarke Judea Pearl Shafi Goldwasser Silvio Micali 9 Richard Hamming Marvin Minskey John McCarthy Edsger Dijkstra Donald Knuth Michael Rabin John Backus Robert Floyd Tony Hoare 0	3 6

ACM Mid-Central Programming Competition 2013

Problem B: Round Robin

Source file: `round.{c, cpp, java}`
Input file: `round.in`

Suppose that N players sit in order and take turns in a game, with the first person following the last person, to continue in cyclic order. While doing so, each player keeps track of the number of turns he or she has taken. The game consists of rounds, and in each round T turns are taken. After a round, the player who just had a turn is eliminated from the game. If the remaining players have all had the same number of turns, the game ends. Otherwise, they continue with another round of T moves, starting with the player just after the one who was most recently eliminated.

As an example, assume we begin a game with N=5 and T=17, labeling the players in order as A, B, C, D, and E, with all counts initially zero.

Player	A	B	C	D	E
Count	0	0	0	0	0

Beginning with A, 17 turns are taken. B will have taken the last of those turn, leaving our counts as follows:

Player	A	B	C	D	E
Count	4	4	3	3	3

Suppose that after *every* 17 turns, the player who just had a turn is eliminated from the game. All remaining players then compare their counts. If all of those counts are equal, everyone has had a fair number of turns and the game is considered completed. Otherwise, they continue with another round of 17 moves starting with the player just after the one who was most recently eliminated.

Continuing with our example, B will leave the game, and the next turn will be for C.

Player	A	C	D	E
Count	4	3	3	3

After 17 more turns starting with C, we find that A, D and E received 4 turns, while C received 5 turns, including the last:

Player	A	C	D	E
Count	8	8	7	7

Then C leaves, and since the remaining counts are not all the same, a new round beings with D having the next turn.

Player	A	D	E
Count	8	7	7

The next 17 turns start with D and end with E. A adds 5 turns, while D and E add 6:

Player	A	D	E
Count	13	13	13

Then E leaves.

Player	A	D
Count	13	13

At this point, notice that the two remaining players have the same count of 13. Therefore, the game ends. (We note that E's count was irrelevant to the decision to end the game.)

Input: The input will contain one or more datasets. Each dataset will be described with a single line containing two integers, N and T, where N ($2 \le N \le 100$) is the initial number of players, and T ($2 \le T \le 100$) is the number of turns after which the player with the most recently completed turn leaves. Following the last dataset is a line containing only 0.

Output: There is one line of output for each dataset, containing two numbers, p and c. At the time the game ends p is the number of players that remain in the game and c is the common count they all have.

Example input:	**Example output:**
5 17 7 10 90 2 0	2 13 5 3 45 1

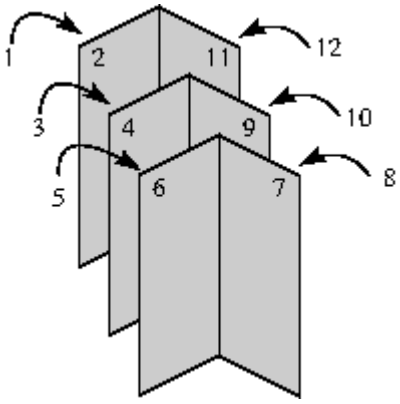
ACM Mid-Central Programming Competition 2013

Problem C: Missing Pages

Source file: `missing.{c, cpp, java}`
Input file: `missing.in`

Long ago, there were periodicals called *newspapers*, and these newspapers were printed on *paper*, and people used to *read* them, and perhaps even share them. One unfortunate thing about this form of media is that every so often, someone would like an article so much, they would take it with them, leaving the rest of the newspaper behind for others to enjoy. Unfortunately, because of the way that paper was folded, not only would the page with that article be gone, so would the page on the reverse side and also two other pages that were physically on the same sheet of folded paper.

For this problem we assume the classic approach is used for folding paper to make a booklet that has a number of pages that is a multiple of four. As an example, a newspaper with 12 pages would be made of three sheets of paper (see figure below). One sheet would have pages 1 and 12 printed on one side, and pages 2 and 11 printed on the other. Another piece of paper would have pages 3 and 10 printed on one side and 4 and 9 printed on the other. The third sheet would have pages 5, 6, 7, and 8.



When one numbered page is taken from the newspaper, the question is what other pages disappear.

Input: Each test case will be described with two integers N and P, on a line, where $4 \le N \le 1000$ is a multiple of four that designates the length of the newspaper in terms of numbered pages, and $1 \le P \le N$ is a page that has been taken. The end of the input is designated by a line containing only the value 0.

Output: For each case, output, in increasing order, the page numbers for the other three pages that will be missing.

Example input:	Example output:
12 2 12 9 8 3 0	1 11 12 3 4 10 4 5 6

ACM Mid-Central Programming Competition 2013

(this page intentionally blank for printing)

Problem D: Probability Paradox

Source file: `paradox.{c, cpp, java}`
Input file: `paradox.in`

This problem considers repeated tosses of a fair coin. Each outcome, H or T, has probability 1/2. Any specific sequence of tosses of the same length, like HHH or THH, has the same probability (for example, 1/8 for a sequence of length 3).

Now consider the following two-player game. Each player chooses a distinct pattern of possible coin flips having a common fixed length. For example, the first player might predict HHH while the second predicts THH. The game then begins with both players observing the same coin as it is repeatedly flipped, until one of them witnesses their pattern. For example, if the sequence of observed flips begins HHTHTTHH... the second player in our example wins the game, having just witnessed the pattern THH.

The question that interests us is, given the two players' patterns, how likely is it that the first player wins the game? Because we are flipping a fair coin, many would assume that the patterns are irrelevant and that each player has probability 1/2 of winning the game. However, this is not the case, leading to what is known as the **Probability Paradox**.

For some patterns, it will be that the first player wins with probability precisely 1/2. For example, by symmetry, the patterns TT and HH have equal chances of occurring first.

However, consider again our original example in which the first player chooses HHH and the second player chooses THH. For this particular match-up, the only way that the first player can win is if each of the first three tosses are H. For if the *earliest* HHH were to come somewhere other than at the beginning of the game, the pattern could be represented as

...**?**HHH

where "..." means a possibly empty earliest part of the sequence, and "?" refers to the toss immediately before the HHH. The "?" can not refer to an H, as in

...**H**HHH

because there would have been an earlier HHH that ended the game, the underlined part of ...HHHH. Yet if the preceding toss were a T, as in

...**T**HHH.

then the second player would have already won, having observed pattern THH at the underlined ...THHH. Therefore, when considering pattern HHH vs. THH, the first player wins if and only if the first three flips are H, an event that happens with probability 1/8.

As one more example, if the first player chooses TTH and the second chooses THH, the first player will win with probability 2/3. Your job is to write a program that computes such a probability.

Input: The input will contain one or more datasets, each on a single line. Each dataset will consist of two equal-length yet distinct patterns using only characters H and T. The common pattern length will be in the range from 1 to 9, inclusive. There is a space between the two patterns. The input ends with a line containing only "$".

Output: For each data set, output a single line with the *exact* probability that the first sequence precedes the second in a random sequence of fair coin tosses. The probability must be stated as a rational number, reduced to lowest terms, with a "/" between the numerator and denominator. Because each player has a nonzero chance of winning, this probability will always be strictly between 0 and 1.

Warning: The numerator and denominators for all of the final probabilities can be expressed as 32-bit integers. However, depending on your approach, you may need 64-bit integers (type **`long`** in Java or **`long long`** in C++) for some intermediate calculations, and even then you must be careful.

Example input:	Example output:
TT HH HHH THH TTH THH THHTH HTHTT HTTHHHTHT THHHTHTHH $	1/2 1/8 2/3 15/28 259/452

ACM Mid-Central Programming Competition 2013

Problem E: Letter Cubes

Source file: `cubes.{c, cpp, java}`
Input file: `cubes.in`

This problem is based on a puzzle by Randall L. Whipkey.

In the game of Letter Cubes, there are a set of cubes, with each face of each cube having a letter of the alphabet, such that no letter appears more than once within the entire set. The maximum number of cubes is 4, allowing for up to 24 of the 26 letters of the alphabet to occur.

Words are formed by rearranging and turning the cubes so that the top letters of all the cubes together spell a word. The 13 words below have been made using a particular set of cubes.

CLIP
CLOG
CONE
DISH
FAZE
FURL
MARE
MOCK
QUIP
STEW
TONY
VICE
WARD

Only 23 distinct letters were used in the above words, so we will tell you the extra information that a B is included on one cube. Can you now determine the letters on each cube? For the above set of words, there is indeed a unique set of cubes. We will state this solution in canonical form as

ABCHTU DEKLQY FGIMNW OPRSVZ

Note that the letters on each individual cube are stated as a string of characters in alphabetical order, and the four 6-letter strings representing the four cubes are also listed in alphabetical order.

A simpler example relies on two cubes, forming the following 11 two-character strings (although the puzzles are more fun when the strings are actual words, they do not need to be):

PI
MU
HO
WE
WO
BE
MA
HI
RE
AB
PY

The only solution for the two cubes forming these strings is

AEIOUY BHMPRW

The same two cubes could be determined without the last pair PY being listed, as long as you were told that there was a Y on one cube. Your job is to make similar deductions.

Input: The input will contain from 1 to 20 datasets. The first line of each dataset will include a positive integer n ($6 \le n \le 30$) and a character c, described below. The next n lines will each contain a string of uppercase letters. Each string will be the same length, call it k, with $2 \le k \le 4$. Following the last dataset is a line containing only 0.

Returning to the issue of the special character, c, on the first input line for each dataset, there will be two cases to consider. Recall that the implicit set of k cubes must use $6*k$ distinct letters on their collective faces If all $6*k$ of those letters appear within the set of strings, then the character c on the first line of input is a hyphen, '-'. Otherwise, the strings have been chosen so that only one letter on the cubes does not appear. In this case, the character c on the first line of input will be that undisplayed letter. (For example, the B in our opening puzzle.)

Output: There is one line of output for each dataset, containing a 6-letter string for each cube, showing the letters on the faces of that cube. Each of those strings should have its letters in alphabetical order, and the set of strings should be given in alphabetical order with respect to each other, with one space between each pair. *We have chosen datasets so that each has a unique solution.*

Example input:	Example output:
13 B CLIP CLOG CONE DISH FAZE FURL MARE MOCK QUIP STEW TONY VICE WARD 11 - PI MU HO WE WO BE MA HI RE AB PY 10 Y PI MU HO WE WO BE MA HI RE AB 0	ABCHTU DEKLQY FGIMNW OPRSVZ AEIOUY BHMPRW AEIOUY BHMPRW

ACM Mid-Central Programming Competition 2013

(this page intentionally blank for printing)

Problem F: Digit Sum

Source file: `digitsum.{c, cpp, java}`
Input file: `digitsum.in`

When Grace was in third grade, her elementary school teacher assigned her the following problem:

> *What is the smallest possible sum of two numbers that together use the numerals 1, 2, 7, 8, and 9?*

Grace figured out that the answer to this problem is 207 (for example, as 78 + 129), but when the teacher assigned four pages of similar problems as homework, Grace got bored. It turns out that Grace was a rather advanced third grader, so she decided that it would be more fun to write a computer program to solve such problems. Surely you can do the same!

Input: Each problem is described on a single line. The line begins with an integer N, such that $2 \le N \le 14$, designating the number of numerals included in the problem. Following that are those N numerals. There will always be at least 2 numerals that are nonzero. The end of the input is designated by a line containing only the value 0.

Output: For each case, output a line with the minimum sum S that can be achieved. Please keep in mind that by standard convention, the numeral *0* cannot appear as the first digit of either summand.

Example input:	**Example output:**
5 1 2 7 8 9 6 3 4 2 2 2 2 9 0 1 2 3 4 0 1 2 3 0	207 447 11257

ACM Mid-Central Programming Competition 2013

(this page intentionally blank for printing)

Problem G: Cash Cow

Source file: `cashcow.{c, cpp, java}`
Input file: `cashcow.in`

Years before Candy Crush became the wildly popular game that may lead developer Saga to a multi-billion dollar IPO, there was an online game named Cash Cow, which remains part of the Webkinz platform.

This single-player game has a board with 12 rows and 10 columns, as shown in Figure 1. We label the rows 1 through 12, starting at the bottom, and the columns *a* through *j*, starting at the left. Each grid location can either have a colored circle or be empty. (We use uppercase characters to denote distinct colors, for example with B=blue, R=red, and Y=yellow.) On each turn, the player clicks on a circle. The computer determines the largest "cluster" to which that circle belongs, where a cluster is defined to include the initial circle, any of its immediate horizontal and vertical neighbors with matching color, those circles' neighbors with matching colors, and so forth. For example, if a user were to click on the blue circle at cell (h10) in Figure 1, its cluster consists of those cells shown with empty circles in Figure 2.

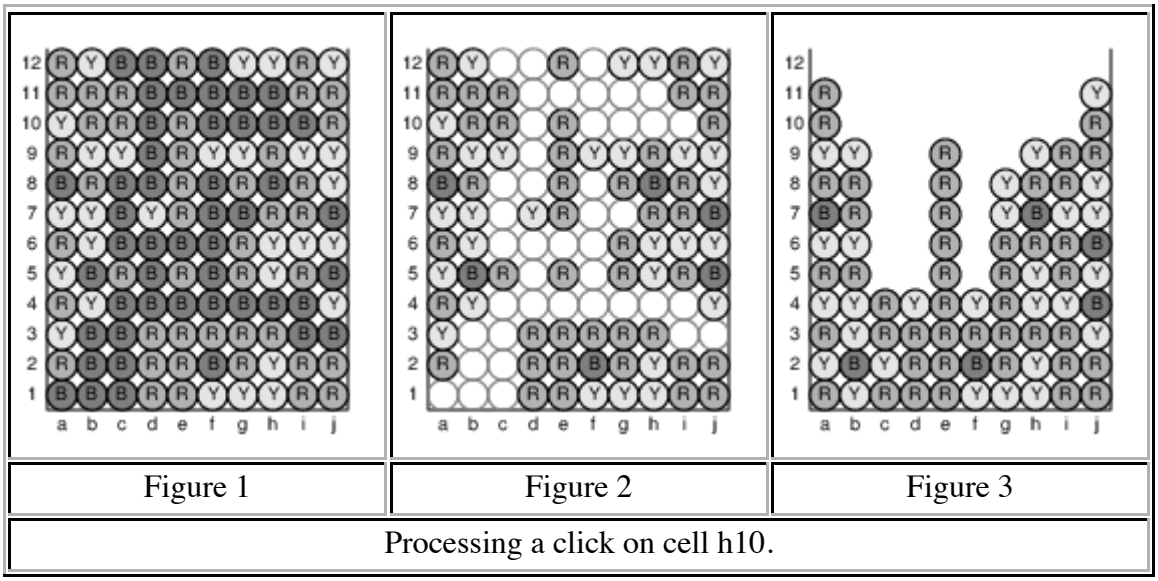


Figure 1	Figure 2	Figure 3
Processing a click on cell h10.		

The player's turn is processed as follows. If the indicated grid cell belongs to a cluster of only one or two circles (or if there is no circle at that cell), the turn is wasted. Otherwise, with a cluster of 3 or more circles, all circles in the cluster are removed from the board. Remaining circles are then compacted as follows:

1. Circles fall vertically, to fill in any holes in their column.
2. If one or more columns have become empty, all remaining columns slide leftward (with each nonempty column remaining intact), such that they are packed against the left edge of the board.

For example, Figure 3 shows the board after the cluster of Figure 2 was removed after the click on (h10).

As another example, Figure 4 below, portrays the processing of a subsequent click on cell (j1). During that turn, column (e) becomes empty, and the resulting columns (f) through (j) slide to become columns (e) through (i), respectively. Figure 5 provides one further example in which several columns are compacted.

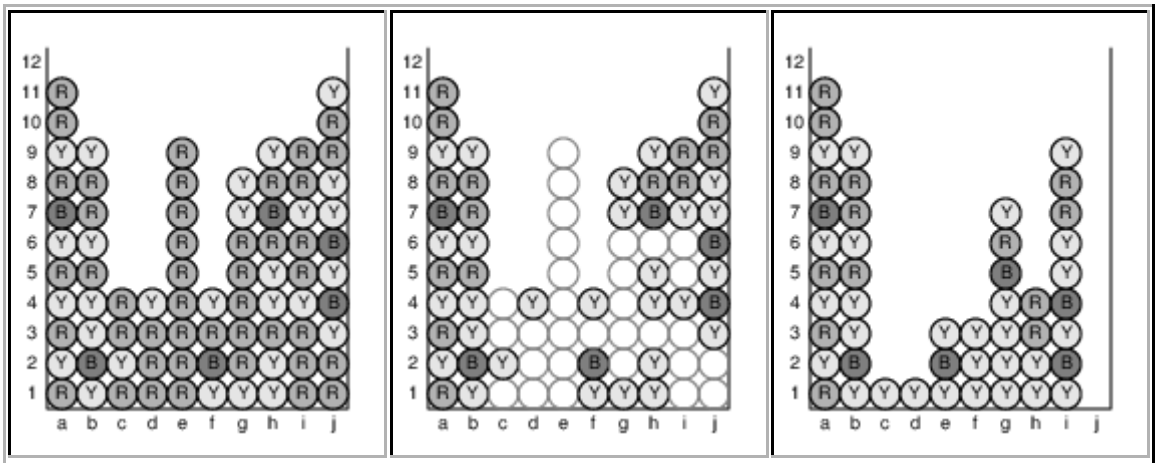


Figure 4: Processing a click on cell j1.

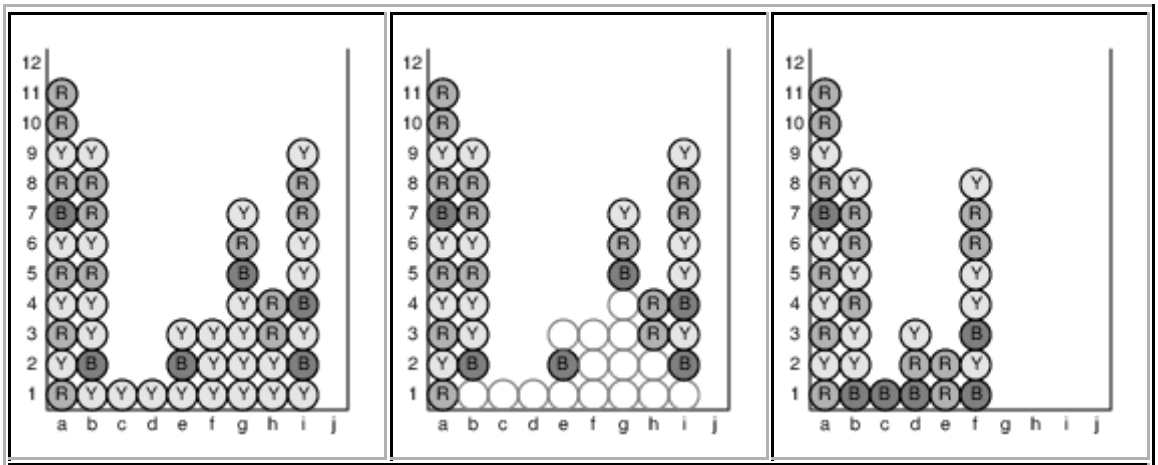


Figure 5: Processing a click on cell g2.

Input: The input will consist of multiple games, each played with a new board. For each game, the input begins with a number T that denotes the number of turns that the player will be making, with $1 \le T \le 20$. Following that will be an initial board configuration, which always has 12 rows and 10 columns per row, with uppercase letters used to denote distinct colors. There will never be empty cells within the initial board. Following the presentation of the initial board will be T additional lines of input, each designating a cell of the grid; we rely on the coordinate system illustrated in the above figures, with a lowercase letter, from `a` to `j`, denoting a column and a number from 1 to 12 that denotes a row. We note that if a player clicks on a grid cell that does not currently have any circle, that turn is simply wasted.

The end of the entire input will be designated by a line with the number 0.

Output: For each game, output a single line designating the the number of circles that remain on the board after all of the player's turns are processed.

Example input:	Example output:
3 RYBBRBYYRY RRRBBBBBRR YRRBRBBBBR RYYBRYYRYY BRBBRBRBRY YYBYRBBRRB RYBBBBRYYY YBRBRBRYRB RYBBBBBBBY YBBRRRRRBB RBBRRBRYRR BBBRRYYYRR h 10 j 1 g 2 3 YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYBYYBBBBB YYBYYBBBBB c 2 c 12 g 1 2 YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYBYYBBBBB YYBYYBBBBB g 1 c 12 0	33 62 2

(this page intentionally blank for printing)

Problem H: Sort Me

Source file: `sortme.{c, cpp, java}`
Input file: `sortme.in`

We know the normal alphabetical order of the English alphabet, and we can then sort words or other letter sequences. For instance these words are sorted:

ANTLER
ANY
COW
HILL
HOW
HOWEVER
WHATEVER
ZONE

The standard rules for sorting letter sequences are used:

1. The first letters are in alphabetical order.
2. Among strings with the same prefix, like the prefix AN in ANTLER and ANY, they are ordered by the first character that is different, T or Y here.
3. One whole string may be a prefix of another string, like HOW and HOWEVER. In this case the longer sequence comes after the shorter one.

The Gorellians, at the far end of our galaxy, have discovered various samples of English text from our electronic transmissions, but they did not find the order of our alphabet. Being a very organized and orderly species, they want to have a way of ordering words, even in the strange symbols of English. Hence they must determine their own order. Unfortunately they cannot agree, and every Gorellian year, they argue and settle on a new order.

For instance, if they agree on the alphabetical order

UVWXYZNOPQRSTHIJKLMABCDEFG

then the words above would be sorted as

WHATEVER
ZONE
HOW
HOWEVER
HILL
ANY
ANTLER
COW

The first letters of the words are in *their* alphabetical order. Where words have the same prefix, the first differing letter determines the order, so the order goes ANY, then ANTLER, since Y is before T in *their* choice of alphabet. Still HOWEVER comes after HOW, since HOW is a prefix of HOWEVER.

Dealing with the different alphabetical orders each year by hand (or tentacle) is tedious. Your job is to implement sorting with the English letters in a specified sequence.

Input: The input will contain one or more datasets. Each dataset will start with a line containing an integer n and a string s, where s is a permutation of the English uppercase alphabet, used as the Gorellians' alphabet in the coming year. The next n lines ($1 \le n \le 20$) will each contain one non-empty string of letters. The length of each string will be no more than 30. Following the last dataset is a line containing only 0.

Output: The first line of output of each dataset will contain "year " followed by the number of the dataset, starting from 1. The remaining n lines are the n input strings sorted assuming the alphabet has the order in s.

Example input:	Example output:
8 UVWXYZNOPQRSTHIJKLMABCDEFG ANTLER ANY COW HILL HOW HOWEVER WHATEVER ZONE 5 ZYXWVUTSRQPONMLKJIHGFEDCBA GO ALL ACM TEAMS GO 10 ZOTFISENWABCDGHJKLMPQRUVXY THREE ONE NINE FIVE SEVEN ZERO TWO FOUR EIGHT SIX 0	year 1 WHATEVER ZONE HOW HOWEVER HILL ANY ANTLER COW year 2 TEAMS GO GO ALL ACM year 3 ZERO ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE

ACM Mid-Central Programming Competition 2013