

**2011-0916 UM Online Practice Programming Contest
September 16, 2011**

**Sponsored by:
University of Michigan**

Rules:

1. There are a few questions to be tried out any time during the practice contest.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. The allowed programming languages are C, C++ and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
7. All communication with the judges will be handled by the PC2 environment.
8. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: To and Fro

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is “There’s no place like home on a snowy night” and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character ‘x’ to pad the message out to make a rectangle, although he could have used any letter.

Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2 . . . 20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Sample Input

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Sample Output

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

Problem B: Team Rankings

It's preseason and the local newspaper wants to publish a preseason ranking of the teams in the local amateur basketball league. The teams are the Ants, the Buckets, the Cats, the Dribblers, and the Elephants. When Scoop McGee, sports editor of the paper, gets the rankings from the selected local experts down at the hardware store, he's dismayed to find that there doesn't appear to be total agreement and so he's wondering what ranking to publish that would most accurately reflect the rankings he got from the experts. He's found that finding the *median ranking* from among all possible rankings is one way to go.

The median ranking is computed as follows: Given any two rankings, for instance **ACDBE** and **ABCDE**, the *distance* between the two rankings is defined as the total number of pairs of teams that are given different relative orderings. In our example, the pair **B**, **C** is given a different ordering by the two rankings. (The first ranking has **C** above **B** while the second ranking has the opposite.) The only other pair that the two rankings disagree on is **B**, **D**; thus, the distance between these two rankings is 2. The median ranking of a set of rankings is that ranking whose sum of distances to all the given rankings is minimal. (Note we could have more than one median ranking.) The median ranking may or may not be one of the given rankings.

Suppose there are 4 voters that have given the rankings: **ABDCE**, **BACDE**, **ABCED** and **ACBDE**. Consider two candidate median rankings **ABCDE** and **CDEAB**. The sum of distances from the ranking **ABCDE** to the four voted rankings is $1 + 1 + 1 + 1 = 4$. We'll call this sum the *value* of the ranking **ABCDE**. The value of the ranking **CDEAB** is $7 + 7 + 7 + 5 = 26$.

It turns out that **ABCDE** is in fact the median ranking with a value of 4.

Input

There will be multiple input sets. Input for each set is a positive integer n on a line by itself, followed by n lines (n no more than 100), each containing a permutation of the letters **A**, **B**, **C**, **D** and **E**, left-justified with no spaces. The final input set is followed by a line containing a 0, indicating end of input.

Output

Output for each input set should be one line of the form:

ranking is the median ranking with value *value*.

Of course *ranking* should be replaced by the correct ranking and *value* with the correct value. If there is more than one median ranking, you should output the one which comes first alphabetically.

Sample Input

```
4
ABDCE
BACDE
ABCED
ACBDE
0
```

Sample Output

```
ABCDE is the median ranking with value 4.
```

Problem C: Sleepwalking through the Mansion

Restless from organizing programming contests, Mark has taken up sleep walking. In his sleep he roams the corridors of his mansion. We can think of his home as being composed of rooms connected by corridors. When Mark enters a room he always wanders through the most interesting corridor connected to that room.

Aware of his behavior, Mark has calculated from each room what the next room he will visit will be. Now Mark is interested in calculating how many different rooms on average he will visit during each of his sleep walks. Assume Mark starts in each room with equal probability.

Input

Input will consist of several test cases. The first line of each case will contain a single integer N ($1 \leq N \leq 100,000$), the number of rooms in the mansion. The next N lines will contain a description of where Mark will head from each room. The first line will indicate where Mark will head from room 0, the second where Mark will head from room 1, and so on. The final input set is followed by a line containing a 0, indicating the end of input.

Output

For each test case, output a single number formatted to 6 decimal points indicating the average number of rooms Mark will visit.

Sample Input

```
3
1
2
1
0
```

Sample Output

```
2.333333
```

Sample Explanation

From room 0 Mark will visit 0, 1, 2, 1, 2, ... (3 unique rooms)

From room 1 Mark will visit 1, 2, 1, 2, 1, ... (2 unique rooms)

From room 2 Mark will visit 2, 1, 2, 1, 2, ... (2 unique rooms)

Problem D: Hit or Miss

One very simple type of solitaire game known as “Hit or Miss” (also known as “Frustration,” “Harvest,” “Roll-Call,” “Talkative”, and “Treize”) is played as follows: take a standard deck of 52 playing cards — four sets of cards numbered 1 through 13 (suits do not matter in this game) which have been shuffled — and start counting through the deck 1, 2, 3, . . . , and so on. When your count reaches 13, start over at 1. Each time you count, look at the top card of the deck and do one of two things: if the number you count matches the value of the top card, discard it from the deck; if it does not match it, move that card to the bottom of the deck. You win the game if you are able to remove all cards from the deck (which may take a very long time).

A version of this game can be devised for two or more players. The first player starts as before with a 52 card deck, while the other players have no cards initially. As the first player removes cards from her deck, she gives them to the second player, who then starts playing the same game, starting at count 1. When that player gets a match, he passes his card to the third player, and so on. The last player discards matches rather than passing them to player 1. All players who have cards to play with perform the following 2-step cycle of moves in lockstep:

1. Each player says his or her current count value and checks for a match. If there is no match, the top card is moved to the bottom of the deck; otherwise it is passed to the next player (or discarded if this is the last player).
2. Each player except the first takes a passed card (if there is one) and places it at the bottom of his or her deck.

These rules are repeated over and over until either the game is won (all the cards are discarded by the last player) or an unwinnable position is reached. If any player ever runs out of cards, he waits until he is passed a card and resumes his count from where he left off (e.g., if player 3 passes his last card on a count of 7, he waits until he receives a card from player 2 and resumes his count with 8 at the beginning of the next 2-step cycle).

Input

Input will consist of multiple input sets. The first line of the file will contain a single positive integer n indicating the number of input sets in the file. Each input set will be a single line containing 53 integers: the first integer will indicate the number of players in the game and the remaining 52 values will be the initial layout of the cards in the deck, topmost card first. These values will all lie in the range 1 . . . 13, and the number of players will lie in the range 1 . . . 10.

Output

For each input set, output the input set number (as shown below, starting with 1) and either the phrase “unwinnable” or a list showing the last card discarded by each player. Use a single blank to separate all outputs.

Sample Input

NOTE: Each sample input set below is split across multiple lines in order to fit on the page – in the actual file each set will be on a single line.

```
2
4 1 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13
  1 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13
4 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13 1
  2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13 1
```

Sample Output

```
Case 1: 13 13 13 13
Case 2: unwinnable
```

Problem E: Anti-prime Sequences

Given a sequence of consecutive integers $n, n+1, n+2, \dots, m$, an *anti-prime sequence* is a rearrangement of these integers so that each adjacent pair of integers sums to a composite (non-prime) number. For example, if $n = 1$ and $m = 10$, one such anti-prime sequence is 1, 3, 5, 4, 2, 6, 9, 7, 8, 10. This is also the lexicographically first such sequence.

We can extend the definition by defining a degree d anti-prime sequence as one where all consecutive subsequences of length $2, 3, \dots, d$ sum to a composite number. The sequence above is a degree 2 anti-prime sequence, but not a degree 3, since the subsequence 5, 4, 2 sums to 11. The lexicographically first degree 3 anti-prime sequence for these numbers is 1, 3, 5, 4, 6, 2, 10, 8, 7, 9.

Input

Input will consist of multiple input sets. Each set will consist of three integers, n , m , and d on a single line. The values of n , m and d will satisfy $1 \leq n < m \leq 1000$, and $2 \leq d \leq 10$. The line 0 0 0 will indicate end of input and should not be processed.

Output

For each input set, output a single line consisting of a comma-separated list of integers forming a degree d anti-prime sequence (do not insert any spaces and do not split the output over multiple lines). In the case where more than one anti-prime sequence exists, print the lexicographically first one (i.e., output the one with the lowest first value; in case of a tie, the lowest second value, etc.). In the case where no anti-prime sequence exists, output

No anti-prime sequence exists.

Sample Input

```
1 10 2
1 10 3
1 10 5
40 60 7
0 0 0
```

Sample Output

```
1,3,5,4,2,6,9,7,8,10
1,3,5,4,6,2,10,8,7,9
No anti-prime sequence exists.
40,41,43,42,44,46,45,47,48,50,55,53,52,60,56,49,51,59,58,57,54
```