11th Annual Manitoba Programming Contest
September 30, 2017

General Instructions:

1. This document is printed double-sided. Please read both sides of each page.
2. Submit solutions using the PC^2 software.
3. All input should be read from standard input.
4. All output should be written to standard output.
5. Each problem has a five (5) second time limit for CPU time when executed on the judging data.

UNIVERSITY OF MANITOBA

Problem 1 – Different, In Order

Given a binary string S of zeroes and ones, you would like to construct strings that are different from S in exactly k positions, for some value of k. Additionally, would like to know which strings are the first, second, third, etc., in lexicographic (AKA "phonebook", "alphabetical") order, among all the strings that differ from S in exactly k positions.

For instance, consider the input string S=000010101. If you want a string that is different in exactly k=4 positions, several strings are possible: 111110101, 011100101, etc. However, the first three strings in lexicographical order that differ in 4 positions are 000000010, 000001000 and 000001011.

Given a binary string S, a value of k and a position p, calculate the p-th string that differs in exactly k positions from S. The values of S, k and p are chosen to guarantee that such a string exists.

Input

Each line of input gives a string S, an integer k and an integer p. The length of S is at most 64, k satisfies $0 \leq k \leq 64$ and p satisfies $1 \leq p \leq 2^{64}$.

Output

For each input case, output the unique string of zeroes and ones that differs from S in exactly k positions and is the p-th string in lexicographical order among all such strings.

Sample Input	Sample Output
000010101 4 1 000010101 4 2 000010101 4 3	000000010 000001000 000001011

Problem 2 – WiFi

You have been awarded a contract to help install a WiFi network in a newly constructed building commissioned by a Three Letter Agency™ (TLA).
This TLA is pretty cautious, and has several rules that must be met in their buildings:

1. The walls and floors must be *at least* 10 inches thick to prevent WiFi signals from entering or leaving the building,
2. Construction workers must wear tin foil under their hard hats (at all times).
3. The 10 inch thick wall rule extends even to *interior* walls (you never know who's listening).

Your job is to help map out how much coverage a wireless access point can provide, given that the signal is going to be blocked by interior walls. Each wireless access point has a broadcast radius. The contractor building the building has provided you with floor plans:

```
........#...
........#...
........#...
........#...
.......w#...
#########...
............
............
```

The symbols on the floor plan indicate:

1. `.` is an open floor space
2. `#` is a 10 inch thick concrete wall
3. `w` is the proposed location of the wireless access point

Each location on the floor plan represents a 10-inch square. Wherever a wall is marked, it fills an entire floor plan location. The signal can be picked up in any location on the floor plan where a straight line between the centre of the location and the centre of the wireless access point's location is not blocked by any part of a wall, and the distance is equal to the radius of the access point or less.

Given the radius of how far the wireless access point is capable of transmitting, your job is to mark on the floor plan where the wireless signal can be picked up. For example, given the above floor plan and a transmitting radius of 3 (i.e., 30 inches), the floor plan should look like:

```
--------#---
-------+#---
-----+++#---
-----+++#---
----+++w#---
#########---
------------
------------
```

Input

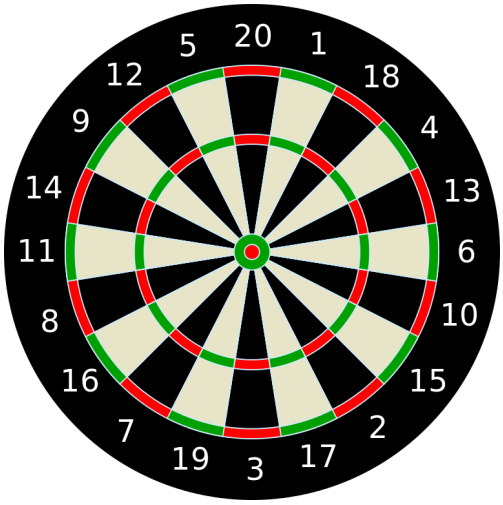
The first line of input is a single integer indicating how many floor plans you're going to be processing. Each floor plan starts with a line with three integers: the radius of the access point `r`, and the width `w` and height `h` of the floor plan. The next `h` lines are the floor plan (with the same symbol set described above). You may assume that the `w`,`r`, and `h` are non-zero and positive.

Output

For each floor plan, output "`FLOORPLAN #X:`" on the first line, where `X` is the case number, starting at 0. After this line, print out a floor plan with the same dimensions and locations of walls and access points. The floor plan should show where the WiFi signal can be picked up (+) and where it cannot be picked up (-).

Sample Input	Sample Output
2 3 12 8 ....#... ....#... ....#... ....#... ....w#... #########... .... 5 14 9 ....#..... ....#..... ....#..... ....w..... .... #########.....	FLOORPLAN #0: --------#--- -------+#--- -----+++#--- -----+++#--- ----+++w#--- #########--- ------------ ------------ FLOORPLAN #1: ----+---#---+- ----++--#--++- ----+++-#-+++- ---+++++w+++++ ----+++++++++- ----+++++++++- #########++++- ---------+++-- --------------

Problem 3 – Dartistic Vision



Darts is a game played by throwing pointed objects (the "darts") at a round board hanging some distance away. The board is divided into several radial sections (i.e., shaped like a slice of pizza), with scores of 1-20 and the bullseye (worth 50). Additionally, each radial section also has portions that are worth double or triple the base score, and there is a ring worth 25 around the bullseye, called the outer bullseye.

Dart players take turns throwing three darts, and players keep track of their score with the aim of reaching an exact score (501 or 301 points are common). As a result, players need to know which combination of dart throws will get an exact score. This is complicated by the fact that the final throw of the game must be either a double-scoring throw or a bullseye.

Given a score, determine if it is possible to throw exactly three darts to obtain that score, with the final being a double-scoring dart or bullseye. For example, if the target score is 39, one possible answer would be 1-2-D18, where D18 indicates a double score on the 18-point region (i.e., 36 points). If the target score is 156 then one possible score would be T20-T20-D18.

In your answer, any valid combination is acceptable, as long as it ends with a double score or a bullseye. Use T and D to indicate triple and double scores, and use OUTER and BULL to denote the outer and inner bullseyes, respectively. If a score is not possible with three darts, you should note this.

Input

The input consists of one integer on each line. The first line gives the number of cases in the input, `n`. The next `n` lines each give a score to complete in one turn, if possible. Each integer will be greater than zero.

Output

For each input, output, on a single line, exactly three dart scores that give the desired total, with the last score being a double score. Indicate single scores by a number (from 1 to 20), double scores by a capital D and a number (1-20), triple scores by a capital T and a number (1-20), bullseye by BULL and outer bullseye

by OUTER. Separate dart scores with a space. If a score is not possible, write "NOT POSSIBLE".

Sample Input	Sample Output
3 39 156 500	1 2 D18 T20 T20 D18 NOT POSSIBLE

Problem 4 – Trophical Vacation

Living organisms in an environment can be organized into food chains, which are ordered graphs that reflect which organisms eat which other organisms in a particular environment. At the lowest level of the food chains are organisms like plants and algae. Higher levels of a food chain contain predators and the top level contains apex predators like sharks.

The trophic level of an organism is its position in a food chain. Historically, the levels were identified by integer values: level 1 are the plants and algae, plant eaters were at level 2, and if an organism ate level X organisms then it was at level X+1.

However, there is also a fractional trophic level based on the level of organisms that an organism eats, but also the proportion of organisms that it eats. In particular, either an organism does not eat any other organisms, and is given a trophic level of 1, or it eats other organisms, and is given a fractional trophic level that is greater than 1. To calculate this fractional trophic level, suppose an organism Q eats organisms P_1, P_2, …, P_n and that

1. Organism P_i is p_i percent of Q's diet
2. Organism P_i has trophic level T_i

Then the trophic level of Q is $1 + \sum_{i=1}^{n} T_i \frac{p_i}{100}$.

Given a collection of organisms and their diets, calculate the trophic levels of all of them.

Input

The input file consists of several test cases. Each test case begins with a single integer `n` on a line, which represents the number of organisms in the food chain. The next `n` lines give the information about each organism. Each line starts with the organsim's name, a string of at most 8 non-whitespace characters. After the name is information about which organisms that organism eats:

- If the organism does not eat any other organisms, after the name is `-1`.
- Otherwise, there is a sequence of pairs `p Q`, which represents that organism with name `Q` represents `p` % of the current organism's diet. The total of all percentages is 100.

The names in the diet of an organism are all valid names in the current food web, but are not guaranteed to be in order (from left to right in the row) or have appeared already (from top to bottom in the input).

The final line of the file is a zero, which indicates the end of the input. You should not process this number.

Output

For each food chain, output the trophic level of each organism. Start the output with `--CASE X--` where `X` is the current case number (starting from 1). Then for each organism, output a single space, then the name of the organism followed by a colon, another space, and then the trophic level. You should round your final trophic levels to two decimal places.

Sample Input	Sample Output
7 a1 -1 a2 -1 a3 100 a2 a4 50 a1 50 a3 a5 75 a3 25 a4 a6 40 a3 10 a5 50 a4 a7 100 a6 0	--CASE 1-- a1: 1.00 a2: 1.00 a3: 2.00 a4: 2.50 a5: 3.13 a6: 3.36 a7: 4.36

Problem 5 – Sock Factory

You work at a sock factory that produces several different kinds of high quality socks. The socks are foot-specific, so that there are different left and right socks of each type, and each pair that is packaged for sale must have a left and right sock.

On a good day, the sock machines produce one kind of sock at a time, alternating right and left socks, and these socks are fed onto a conveyer belt. Your job at the factory is pair assembler: you pick up left and right socks off the conveyer belt and package them together for sale.

Unfortunately, a bug in the sock machine software has caused the usual sock pattern to be broken and now socks have been placed on the conveyer belt randomly. The conveyer belts have are now full of all different kinds of socks, including both right and left socks.

To fix this, all the pair assemblers have been called in for overtime work to assemble pairs. The only rule that you've been given (to avoid chaos on the conveyer belt) is that you should not wander around the belt looking for pairs. You should only pick up any pairs that are next to each other in line, assemble them, and then continue looking.

It doesn't matter which order (left-right or right-left) the socks appear in on the conveyer belt – each can be made into a pair. All of the pair assemblers are doing this, and as pairs are assembled, new socks are then next to each other on the line. You have enough pair assemblers to assemble all pairs that may appear during the process of pair assembly – no socks that are next to each other at any point during the pair assembly process will be left unpaired.

Given the contents of a conveyer belt, what are the socks that are left on the belt after as many pairs as possible have been assembled?

Input

The first line of input is an integer `n`, giving the number of conveyer belts in the factory. Each of the remaining `n` lines gives the contents of one conveyer belt. The belts are given as a list of a positive number of socks, in order, that appear on the conveyer belt. Socks are represented by a letter from `a` to `z` (so the factory produces at most 26 different types of socks). Left socks are represented by a lower case letter and right socks are represented by an upper-case letter. There are no spaces between the characters.

Output

For each conveyer belt, give the contents after all possible pairs of socks are formed. If several different contents are possible (given different orders of assembling socks), all contents will be accepted as a valid answer. If no socks are left on the conveyer belt, write "EMPTY".

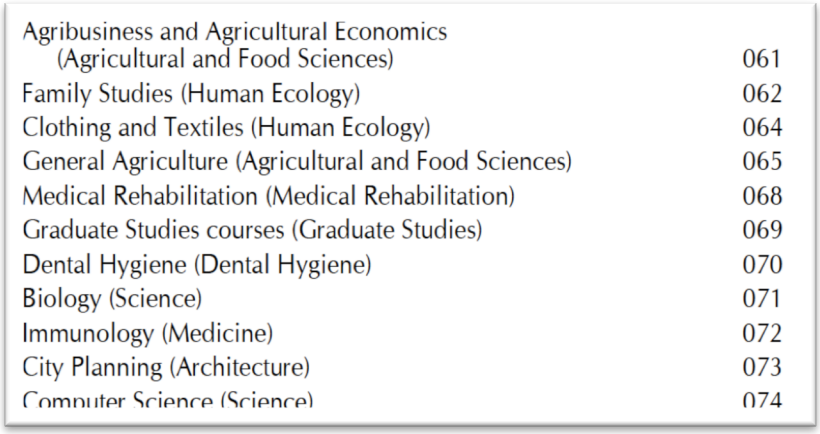
Sample Input	Sample Output
3 aAbBcC aAa AbBaCcDD	EMPTY a DD

Problem 6 – 74.101

At the University of Manitoba, courses didn't have letter codes (like COMP) until around 2006. Before this, a course like COMP 1010 was called 74.101, where 74 was a unique numerical code for computer science. The number 101 was the course number, similar to the 4-digit course code used today.

The course numbering system was changed because aurora, the student registration system that the University of Manitoba began using around 2006, did not handle numerical course codes by default and a custom student registration system was not within the University of Manitoba's budget. Consequently, all course units (like Computer Science) were given a letter code (like COMP) and three digit numbers for individual courses were converted to four digit codes.

The original numerical codes for units were given in a very clear, rational order, as demonstrated in the picture on the right, showing some of the numerical codes from the 2003-2004 University of Manitoba Undergraduate Calendar. Given this selection, it should be very obvious that code 75 was skipped and that code 76 was for the Department of Anthropology.



Agribusiness and Agricultural Economics (Agricultural and Food Sciences)	061
Family Studies (Human Ecology)	062
Clothing and Textiles (Human Ecology)	064
General Agriculture (Agricultural and Food Sciences)	065
Medical Rehabilitation (Medical Rehabilitation)	068
Graduate Studies courses (Graduate Studies)	069
Dental Hygiene (Dental Hygiene)	070
Biology (Science)	071
Immunology (Medicine)	072
City Planning (Architecture)	073
Computer Science (Science)	074

These numerical codes for courses are still used internally at the University of Manitoba to denote a student's major. Some internal views of student records show the numerical code when indicating a student's major department.

The University of Manitoba is interested in tracking majors for different groups of students: students from different countries, students who did well in Calculus, students who live in residence, etc. Given the major information for a collection of students, you have been asked to decide what the two most popular majors are for that set of students. You have been given filtered data that lists the numerical code for each student only. Based on each set of data, there is guaranteed to be two unique majors that are more popular than all others.

Input

The first line of input gives the number of groups of students, which is an integer `n`. The next `n` lines each give a collection of students. The first integer `s` in each line is the number of students in the collection. This is followed by `s` integers,

each of which is an integer between 1 and 200, giving the major of a student. Each numerical code is separated from the next by a single space.

Output

For each collection, output on a single line the two most popular majors in the list of students. List the most popular major first, then the second most popular major.

Sample Input	Sample Output
2 10 74 74 74 73 76 74 1 76 76 73 6 1 1 1 2 2 3	74 76 1 2