University of Manitoba
Open Programming Contest
September 24, 2011

General Instructions:

1. Submit solutions using the PC^2 software.
2. The questions are **not** listed in order of difficulty. Some questions will be easier than others, and these are not necessarily the problems listed first.
3. All input should be read from standard input.

Problem 1 – Locks

In a large building, each door is equipped with a sensor which can tell when someone locks or unlocks a door. Each time either of these events occurs, a letter is written in a log file: L for locked and U for unlocked. The room that is locked or unlocked is not written to the log file. A sample log file would look like ULLUULULULUUULUULLLUU.

This log file is reviewed by the security guards, who are usually concerned with whether any doors in the building have been left unlocked. But recently, the guards have noticed that some door sensors have been malfunctioning. The doors still report all lock events, but have begun to miss some of the unlock events. That is, not every U that should be in the log file actually appears there. So some logs indicate that more doors have been locked than have been unlocked, or an order of locks and unlocks that indicate a malfunction.

A new empty log file is started at the beginning of each day, when all doors are locked. Lock and unlock events must alternate for each door; no door can have two lock events in a row, nor two unlocks in a row.

Given this situation, you are asked to write a log file analyzer which reports suspicious logs to the security guards. Given a log file (a sequence of Ls and Us) you should report one of three messages:

1. “malfunction”: print this when there is a malfunction in one of the mechanisms used to report unlockings.
2. “unlocked door”: use this when there is at least one door which can be proven to be left unlocked.
3. “plausible”: use this when neither “malfunction” or “unlocked door” can be established with certainty.

Input

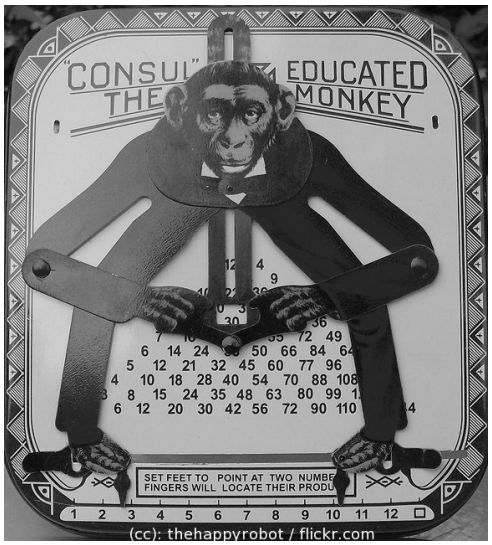
The input is a series of lines, each consisting of a string consisting of U and L only.

Output

For each line of input, output the case number (beginning at 1) as "Case x: " (where x is an integer) and then one of the three outcomes, followed by a period.

Sample Input	Sample Output
LU UL UUL	Case 1: malfunction. Case 2: plausible. Case 3: unlocked door.

Problem 2 – Consul



"Consul" the educated monkey is a monkey that can multiply. No wait. I'm just being told that Consul is in fact a vintage toy monkey that can be manipulated to multiply small numbers.

The toy consists of two moving legs that change the position of a window over a triangle of numbers. When the two legs point at `i` and `j` with $1 < i < j < 12$ the window will read `i*j`.

A picture of Consul is provided. For completeness, the triangle of numbers in the picture looks like this

```
12
11 24
10 22 36
9 20 33 48
8 18 30 44 60
7 16 27 40 55 72
6 14 24 36 50 66 84
5 12 21 32 45 60 77 96
4 10 18 28 40 54 70 88 108
3 8 15 24 35 48 63 80 99 120
2 6 12 20 30 42 56 72 90 110 132
```

Write a program which given an input n outputs the triangle used by Consul to multiply numbers $1 < i < j \leq n$.

Input

The input file consists of several lines. Each line consists of a single positive integer, which is at least two and at most 100.

Output

For each input integer n, output the triangle of numbers described above. Each number should be separated from its neighbours on the same line by a single space.

Sample Input	Sample Output
3 5	3 2 6 5 4 10 3 8 15 2 6 12 20

Problem 3 – Binaristic Sequence

Each infinite sequence of integers defines a characteristic sequence over {0,1}. The characteristic sequence is defined by $s_i = 1$ if i is in the sequence, and $s_i=0$ if i is not in the sequence. For instance, the first few prime numbers are 2,3,5,7,11,13,17,... so its characteristic sequence is

0011010100010100001...

Note that the sequence begins with $i=1$.

We now define another sequence from the characteristic sequence (call it the *binaristic* sequence). Consider blocks of the sequence of increasing length (starting with 1) and then treat these as binary numbers. For instance, for the characteristic sequence of prime numbers, we get

0 01 101 0100 01010 …

which as a binary numbers are

0 1 5 4 10 …

Given an integer n, what is the n-th element of the binaristic sequence for the prime numbers?

Input

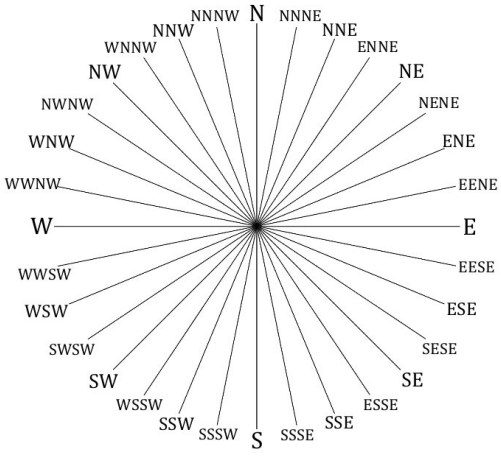
The first line gives the number of test cases in the file. Each subsequent input line consists of a single integer less than or equal to 32.

Output

For each input integer n, give the n-th element of the binaristic sequence for the prime numbers.

Sample Input	Sample Output
3 2 3 4	1 5 4

Problem 4 – Arbitrary Precision Cardinal Points



Everyone knows the cardinal points on a compass: North, East, South and West. These correspond to angles of 0 degrees, 90 degrees, 180 degrees and 270 degrees, respectively (but N is also 360 degrees). Then we have NE (North-east), SE (South-east), SW (South-west) and NW (North-west), which are the midpoints between the respective cardinal points. For instance, SE is at 135 degrees. Finally, the points such as NNE are the midway points between (in this case) NE and N: it is the point North of North-east. NNE corresponds to 22.5 degrees.

In this problem, we extend the concept of cardinal points to arbitrary precision, capable of representing any degree value between 0 and 360 (but not including 360). It is an extension of the previous argument: If X and Y are arbitrary precision labels representing two cardinal points (with no labelled cardinal points written between them), and X is the longer label (i.e., is longer as a string) then the midpoint of the interval from X to Y is labelled by AX where A (one of {N,E,S,W}) is the direction from X to the midpoint.

(There is one exception: if X and Y have the same length, then they must both be one of the four cardinal points (N,E,S,W) and the midpoint is one of NE,SE,SW or NW.)

For example, N is 0 degrees = 360 degrees and NNW is 337.5 degrees. The midpoint of these two points is 348.75 degrees. It is N of NNW, so it would be labelled NNNW (it is not labelled WN, since NNW is the longer of the two endpoint labels). An example of showing the 32 primary cardinal points is given in the picture.

Write a program that is capable of converting a degree measurement d with 0 <= d < 360 into the closest arbitrary precision cardinal point. The numbers will be the degree measurement from cardinal points obtained by dividing intervals at most 10 times.

Input

The first line of input is a number n, which indicates the number of cases in the input. The next n lines each consist of a single floating point number d with 0 ≤ d < 360.

Output

For each line in the file except the first line (the number of cases), output the cardinal point corresponding to the number d on a line by itself.

Sample Input	Sample Output
5 180 22.5 348.75 117.333984375 152.314453125	S NNE NNNW SSESSEESESE ESEEESSESSE

Problem 5 – S-I-R

We can develop a simple model of infectious diseases in a fixed population by dividing the population into three disjoint sets: the set of susceptible people (size S), the set of infected people (size I) and the set of recovering people (size R). Every member of the population is in one of those three groups. The model is updated over a number of time steps.

In the model, once you are infected, you are no longer susceptible and after a period of infection, you move to the recovering group. No one dies from this disease. With this model, there are two parameters:

1. b - the average fraction of people an infected person comes into contact with each time period, expressed as a number between 0 and 1.
2. g - the average recovery rate for an infected person, in number of time periods.

From this, the changes in the population over a time period can be modelled by the changes in the three sets. Let ΔS,ΔI and ΔR be the changes in the sizes of the three sets of people over one time period. Then

$$\Delta S = -bSI$$
$$\Delta I = bSI - gI$$
$$\Delta R = gI$$

Given a initial set of S susceptible individuals and I infected individuals (with no recovering individuals), values for b and g, and a number of time steps t, print the number of susceptible, infected and recovering people after t time steps. (Consider the initial situation to be time step 0.)

Input

The first line of input is a single number N, which is the number of test cases to examine. Each subsequent line begins with a string (the name of the disease) and the four values: S, I, b, g and t, in that order.

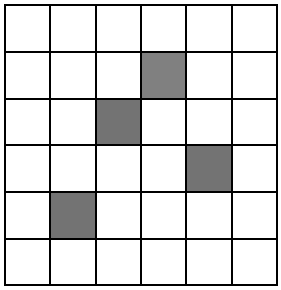
Output

For each disease, output the name of the disease, followed by a colon and space and the values of S, I and R, in that order, each separated by a space. Round all values to the nearest integer. Each output should be on one line.

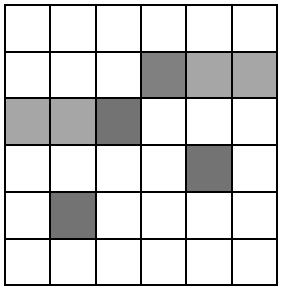
Sample Input	Sample Output
3 measles 1000000 1000 0.0000016 0.001 100 chickenpox 1000000 10 0.0000010 0.01 10 hyperevolutionary-virus 1000 1 0.00000001 0.1 10	measles: 0 912025 88975 chickenpox: 990220 9691 98 hyperevolutionary-virus: 1000 0 1

Problem 6 - Grid

Consider a N-by-N grid where some cells are blocked by obstacles. For example, the following 6-by-6 grid has 4 blocked cell (shaded).



There is a dangerous predator in the top-left cell. The predator can move up, down, left, and right. Your task is place additional obstacles in the grid in order to prevent the predator from reaching the bottom-right cell. No obstacles may be placed in the top and bottom rows. One solution to the above example is given by



Input

The first line of input is a single number, which is the number of test cases to process. Each test case begins with a line with grid size N (N < 60) followed by N strings defining the locations of the initial obstacles. Cell with obstacles are marked with "*" and empty cells are ".".

Output

For each test case, output the **minimum** number of obstacles required to solve the problem. Each output should be on one line.

Sample Input	Sample Output
2 4 .... *... .... 6 ....*.. ..*... ....*. ....*....	3 4