V MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD KONRAD LORENZ

7 de Marzo de 2009

PROBLEMAS

Elaborado por: Hector Florez
Basado de www.acis.org.co, www.acm.org

Problem 1

Calendar Game

Source file name: calen.c, calen.cpp or calen.java
Input: calen.in
Output: standar output

Adam and Eve enter this year's ACM International Collegiate Programming Contest. Last night, they played the Calendar Game, in celebration of this contest. This game consists of the dates from January 1, 1900 to November 4, 2001, the contest day. The game starts by randomly choosing a date from this interval. Then, the players, Adam and Eve, make moves in their turn with Adam moving first: Adam, Eve, Adam, Eve, etc. There is only one rule for moves and it is simple: from a current date, a player in his/her turn can move either to the next calendar date or the same day of the next month. When the next month does not have the same day, the player moves only to the next calendar date. For example, from December 19, 1924, you can move either to December 20, 1924, the next calendar date, or January 19, 1925, the same day of the next month. From January 31, 2001, however, you can move only to February 1, 2001, because February 31, 2001 is invalid.

A player wins the game when he/she exactly reaches the date of November 4, 2001. If a player moves to a date after November 4, 2001, he/she looses the game. Write a program that decides whether, given an initial date, Adam, the first mover, has a winning strategy. For this game, you need to identify leap years, where February has 29 days. In the Gregorian calendar, leap years occur in years exactly divisible by four. So, 1993, 1994, and 1995 are not leap years, while 1992 and 1996 are leap years. Additionally, the years ending with 00 are leap years only if they are divisible by 400. So, 1700, 1800, 1900, 2100, and 2200 are not leap years, while 1600, 2000, and 2400 are leap years.

Input

The input consists of T test cases. The number of test cases (T) is given in the first line of the input file. Each test case is written in a line and corresponds to an initial date. The three integers in a line, YYYY MM DD, represent the date of the DD–th day of MM–th month in the year of YYYY. Remember that initial dates are randomly chosen from the interval between January 1, 1900 and November 4, 2001.

Output

Print exactly one line for each test case. The line should contain the answer "YES" or "NO" to the question of whether Adam has a winning strategy against Eve. Since we have T test cases, your program should output totally T lines of "YES" or "NO".

Sample input	**Sample Output**

3
2001 11 3
2001 11 2
2001 10 3
	YES
	NO
	NO

Problem 2

Decomposition

Source file name: decomposition.c, decomposition.cpp or decomposition.java
Input: standar input
Output: standar output

Definition

70	**2**
35	**5**
7	**7**
1	

Input

There will be multiple cases. Each test case will be contained on one line. Each line contain one number n where 2<n<1000.

Output

For each test case you should print one line of output which contains the answer of the decomposition with the format showed below.

Sample input	**Sample Output**
70 100 150	2^1 * 5^1 * 7^1 2^2 * 5^2 2^1 * 3^1 * 5^2

Problem 3

Copying DNA

Source file name: dna.c, dna.cpp or dna.java
Input: dna.in
Output: standar output

Evolution is a seemingly random process which works in a way which resembles certain approaches we use to get approximate solutions to hard combinatorial problems. You are now to do something completely different.

Given a DNA string S from the alphabet {A,C,G,T}, find the minimal number of copy operations needed to create another string T. You may reverse the strings you copy, and copy both from S and the pieces of your partial T. You may put these pieces together at any time. You may only copy contiguous parts of your partial T, and all copied strings must be used in your final T.

Example: From S = "ACTG" create T = "GTACTATTAAT"

1. Get GT	by copying and reversing "TG" from S.
2. Get GTAC	by copying "AC" from S.
3. Get GTAC...TA	by copying "TA" from the partial T.
4. Get GTAC...TAAT	by copying and reversing "TA" from the partial T.
5. Get GTACAATTAAT	by copying "AAT" from the partial T.

Input

The first line of input gives a single integer, 1 <= t <= 100, the number of test cases. Then follow, for each test case, a line with the string S of length 1 <= m ,= 18, and a line with the string T of length 1 <= n <= 18.

Output

Output for each test case the minimal number of copy operations needed to create T from S, or "impossible" if it cannot be done.

Sample input	Sample Output
5 ACGT GTAC A C ACGT TGCA ACGT TCGATCGA A AAAAAAAAAAAAAAAAAA	2 impossible 1 4 6

Problem 4

Where Is The Ace?

Source file name: ace.c, ace.cpp or ace.java
Input: ace.in
Output: standar output

You come across an apparently respectable man on the street that makes you a seemingly legitimate proposition: the opportunity to play a game of Where Is The Ace. He has a table with three face-up cards in a row: two jokers and one ace, with the ace in the middle. After you make a small wager, he turns the cards face-down and begins to manipulate the cards, swapping cards two at a time. After he completes the swaps, you are then to guess which card is the ace.

The series of card swaps will be given to you in order as a String swaps, containing only the characters L, R, E, and F. The 4 characters indicate the following moves:

L swap the left and middle cards
R swap the right and middle cards
E swap the cards on the ends (the left and right cards)
F fake swap (no cards actually change position)

You are asked to write a program that calculates the final position of the ace, after all the swaps have been performed.

Input

The first line of the input contains a positive integer N which indicates the number of test cases. The following N lines contain the test cases, one per line. Each test case consists of a single line containing a sequence S of characters L, R, E, and F. The lenght of S is between 0 and 50 inclusive.

Output

For each test case your solution should print, in the order of the test cases, a single line containing: L if the left card is the ace, R if the right card is the ace, and M if the ace is in the middle.

Sample input	**Sample Output**
5 L REL RFRFR FLRELFLRELLFRLFEELFLRFLELRFLRFREFRFLLRFERLFLREFRFL	L M R M L

Problem 5

Perfect Number

Source file name: perfect.c, perfect.cpp or perfect.java
Input: standar input
Output: standar output

In mathematics, a perfect number is defined as a positive integer which is the sum of its proper positive divisors, that is, the sum of the positive divisors excluding the number itself.

The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6.

The next perfect number is 28 = 1 + 2 + 4 + 7 + 14.

Input

The input contains several test cases. Each line contain one number n where 1<n<100000.

Output

For each test case your program must write one line, containing the word “Perfect Number”, indicating a perfect number or the word “No Perfect Number” indicating a number that is not a perfect number.

Sample input	**Sample Output**
6 8 20 28	Perfect Number No Perfect Number No Perfect Number Perfect Number

Problem 6

Generalized Matrioshkas

Source file name: matriosh.c, matriosh.cpp or matriosh.java
Input: matriosh.in
Output: standar output

Vladimir worked for years making matrioshkas, those nesting dolls that certainly represent truly Russian craft. A matrioshka is a doll that may be opened in two halves, so that one finds another doll inside. Then this doll may be opened to find another one inside it. This can be repeated several times, till a final doll -that cannot be opened- is reached.

Recently, Vladimir realized that the idea of nesting dolls might be generalized to nesting toys. Indeed, he has designed toys that contain toys but in a more general sense. One of these toys may be opened in two halves and it may have more than one toy inside it. That is the new feature that Vladimir wants to introduce in his new line of toys.

Vladimir has developed a notation to describe how nesting toys should be constructed. A toy is represented with a positive integer, according to its size. More precisely: if when opening the toy represented by m we find the toys represented by $n_1, n_2, \ldots, n_r$, it must be true that $n_1 + n_2 + \ldots + n_r < m$. And if this is the case, we say that toy m contains directly the toys $n_1, n_2, \ldots, n_r$. It should be clear that toys that may be contained in any of the toys $n_1, n_2, \ldots, n_r$ are not considered as directly contained in the toy m.

A generalized matrioshka is denoted with a non-empty sequence of non zero integers of the form: $a_1\ a_2 \ldots a_N$, such that toy k is represented in the sequence with two integers −k and k, with the negative one occurring in the sequence first that the positive one.

For example, the sequence

−9 − 7 − 2 2 − 3 − 2 − 1 1 2 3 7 9

represents a generalized matrioshka conformed by six toys, namely, 1, 2 (twice), 3, 7 and 9. Note that toy 7 contains directly toys 2 and 3. Note that the first copy of toy 2 occurs left from the second one and that the second copy contains directly a toy 1. It would be wrong to understand that the first −2 and the last 2 should be paired.

On the other hand, the following sequences do not describe generalized matrioshkas:

−9 − 7 − 2 2 − 3 − 1 − 2 2 1 3 7 9

because toy 2 is bigger than toy 1 and cannot be allocated inside it.

−9 − 7 − 2 2 − 3 − 2 − 1 1 2 3 7 − 2 2 9

because 7 and 2 may not be allocated together inside 9.

−9 − 7 − 2 2 − 3 − 1 − 2 3 2 1 7 9

because there is a nesting problem within toy 3.

Your problem is to write a program to help Vladimir telling good designs from bad ones.

Input

The input file contains several test cases, each one of them in a separate line. Each test case is a sequence of non zero integers, each one with an absolute value less than 10^7.

Output

Output texts for each input case are presented in the same order that input is read. For each test case the answer must be a line of the form
:-) Matrioshka!
if the design describes a generalized matrioshka. In other case, the answer should be of the form
:-(Try again.

Sample input	**Sample Output**
-9 -7 -2 2 -3 -2 -1 1 2 3 7 9	:-) Matrioshka!
-9 -7 -2 2 -3 -1 -2 2 1 3 7 9	:-(Try again.
-9 -7 -2 2 -3 -1 -2 3 2 1 7 9	:-(Try again.
-100 -50 -6 6 50 100	:-) Matrioshka!
-100 -50 -6 6 45 100	:-(Try again.
-10 -5 -2 2 5 -4 -3 3 4 10	:-) Matrioshka!
-9 -5 -2 2 5 -4 -3 3 4 9	:-(Try again.