MARATÓN DE PROGRAMACIÓN UFPS 2018

SET DE PROBLEMAS

Universidad Francisco de Paula Santander
Vigilada Mineducación

Programa de Ingeniería de Sistemas
Acreditado de Alta Calidad

SILUX

RED DE PROGRAMACIÓN COMPETITIVA
RPC

Contents

MARATÓN DE PROGRAMACIÓN UFPS 2018

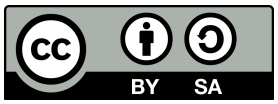
Como cada año, la Maratón de Programación UFPS propone un conjunto de retos para poner a prueba las habilidades en algoritmia, trabajo en grupo y programación de los estudiantes del programa Ingeniería de Sistemas UFPS, y gracias a la Red de Programación Competitiva (RPC) esta competencia llega a diferentes universidades en varios países de Latinoamerica.

El presente conjunto de problemas ha sido escrito por:

- Milton Jesus Vera - UFPS
- Gerson Yesid Lázaro - UFPS
- Angie Melissa Delgado - UFPS
- Hugo Humberto Morales - UTP Pereira
- Javier Eduardo Ojeda - UMSS Bolivia

Y ha sido testeado y validado por

- Milton Jesus Vera - UFPS
- Gerson Yesid Lázaro - UFPS
- Angie Melissa Delgado - UFPS
- Juan Camilo Bages - Uniandes Bogotá



GRUPO DE ESTUDIO EN PROGRAMACIÓN COMPETITIVA

El grupo de estudio en Programación Competitiva hace parte del Semillero de Investigación en Linux y Software Libre (SILUX) y tiene como fin preparar y fortalecer a los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander en algoritmia y programación, en un entorno competitivo que favorece el aprendizaje y el trabajo en equipo.

Integrantes del grupo de estudio han participado en la Maratón Nacional de Programación desde el año 2014, logrando por 4 años consecutivos la clasificación a fase Regional Latinoamericana.

INSTRUCCIONES

Puedes utilizar Java, C, C++ o Python, teniendo en cuenta:

1. Resuelve cada problema en un único archivo. Debes enviar a la plataforma únicamente el archivo .java, .c, .cpp, o .py que contiene la solución.
2. Todas las entradas y salidas deben ser leídas y escritas mediante la entrada estándar (Java: Scanner o BufferedReader) y salida estándar (Java: System.out o PrintWriter).
3. En java, el archivo con la solución debe llamarse tal como se indica en la linea "Source file name" que aparece debajo del título de cada ejercicio. En los otros lenguajes es opcional (puede tener cualquier nombre).
4. En java, la clase principal debe llamarse igual al archivo (Si el Source File Name indica que el archivo debe llamarse example.java, la clase principal debe llamarse example).
5. En java, asegúrate de borrar la linea "package" antes de enviar.
6. Tu código debe leer la entrada tal cual se indica en el problema, e imprimir las respuestas de la misma forma. No imprimas lineas adicionales del tipo "La respuesta es..." si el problema no lo solicita explicitamente.
7. Si su solución es correcta, en unos momentos la plataforma se lo indicará con el texto "YES". En caso contrario, obtendrá un mensaje de error.

REGLAS

1. Gana el equipo que resuelve mas problemas. Entre dos equipos que resuelvan el mismo número de problemas, gana el que los haya resuelto en menos tiempo (ver numeral 2).
2. El tiempo obtenido por un equipo es la suma de los tiempos transcurrido en minutos desde el inicio de la competencia hasta el momento en que se envió cada solución correcta. Habrá una penalización de 20 minutos que se suman al tiempo por cada envío erróneo (esta penalización solo se cuenta si al final el problema fue resuelto).
3. NO se permite el uso de internet durante la competencia. Únicamente se puede acceder a la plataforma en la cual se lleva a cabo la competencia.
4. NO se permite el uso de dispositivos electrónicos durante la competencia.
5. NO se permite la comunicación entre miembros de equipos diferentes. Cada integrante solo puede comunicarse con sus dos compañeros de equipo.
6. Se permite todo tipo de material impreso (libros, fotocopias, apuntes, cuadernos, guías) que el equipo desee utilizar.

PROBLEM A. ATTRACTIVE SUBSEQUENCE

Source file name:	`attractive.c, attractive.cpp, attractive.java, attractive.py`
Input/Output:	`standard`
Author('s):	`Hugo Humberto Morales Peña - UTP Colombia`

You receive a sequence S of non-negative integer numbers. Your task is to calculate the total number of **Attractive Subsequences**. An **Attractive Subsequence** is a subsequence of consecutive elements in S such that the sum of its elements is equal to some given value K. For example, consider the sequence $S = \langle 0,\ 0,\ 25,\ 0,\ 0,\ 25 \rangle$ and the value $K = 25$, there are 12 Attractive Subsequences, these are represented with ordered pairs $(ind_1,\ ind_2)$, where ind_1 is the position of the first element and ind_2 is the position of the last element in the original sequence S. In this representation the Attractive Subsequences are: $(1,\ 3)$, $(1,\ 4)$, $(1,\ 5)$, $(2,\ 3)$, $(2,\ 4)$, $(2,\ 5)$, $(3,\ 3)$, $(3,\ 4)$, $(3,\ 5)$, $(4,\ 6)$, $(5,\ 6)$ y $(6,\ 6)$.

INPUT

The first line in the input contains one integer number T ($1 \leq T \leq 20$), the number of test cases in the input. Each test case contains three lines, the first line contains two positive integer numbers N ($1 \leq N \leq 10^5$) and Q ($1 \leq Q \leq 10^3$), the number of elements in the sequence S and the number of queries respectively. The next line contains exactly N space-separated non-negative integer numbers S_1, S_2, S_3, ..., S_N ($0 \leq S_i \leq 10^3$, for $1 \leq i \leq N$), that is the sequence S. The next line contains exactly Q space-separated positive integer numbers K_1, K_2, K_3, ..., K_Q ($1 \leq K_j \leq 10^7$, for $1 \leq j \leq N$), the queries of the attractive subsequences.

OUTPUT

For each case you must print Q space-separated integer numbers in a single line, one per query, with the number of the attractive subsequences.

EXAMPLE

Input	Output
2 6 2 0 0 25 0 0 25 25 50 7 3 1 6 5 2 3 4 7 7 5 11	12 3 4 2 2

Use fast I/O methods

PROBLEM B. VALENTINE'S DAY

Source file name:	`valentine.c, valentine.cpp, valentine.java, valentine.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Cúcuta`

Valentine's Day has arrived, and Alice has bought a present for Bob: a box of chocolates in the form of a ortohedron (a 6-sided common box with all the right angles) of dimensions $X * Y * Z$. Now Alice wants to wrap it in green paper (her favorite color) but since she is not very good at cutting and pasting, Alice wants to make sure that a single piece of paper reaches you to wrap the entire box.

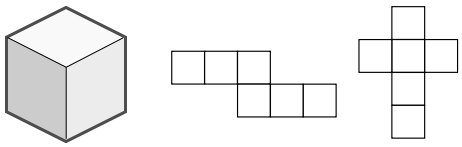


FIGURE 1: BOX OF CHOCOLATES OF $1 * 1 * 1$ AND TWO POSSIBLE PIECES OF PAPER TO WRAP IT (THERE ARE MORE WAYS APART FROM THESE TWO)

With a piece of paper in form of a rectangle of size $A * B$, is it possible to cut a complete piece that wraps around the box? It is not allowed to cut different parts and join them, the piece must be a single fragment.

INPUT

The first line contains an integer T, the number of test cases. Each test case contains a line with 5 integers separated by spaces A, B, X, Y, Z ($1 \leq A, B, C, D, E \leq 10^8$).

OUTPUT

Print one line for each test case, with the string "POSSIBLE" if it is possible to cut a piece of paper completely to wrap the whole box or "IMPOSSIBLE" otherwise.

EXAMPLE

Input	Output
3 5 2 1 1 1 4 3 1 1 1 5 3 2 2 2	POSSIBLE POSSIBLE IMPOSSIBLE

PROBLEM C. CIRCUMSCRIBED RECURSION

Source file name:	`recursion.c, recursion.cpp, recursion.java, recursion.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Colombia`

An inscribed circle in a polygon is the largest possible circle that can be drawn on the inside of a plane figure and it is tangent to each side of the polygon. A circumscribed circle to a polygon passes through all vertices of a plane figure and contains the entire figure in its interior.

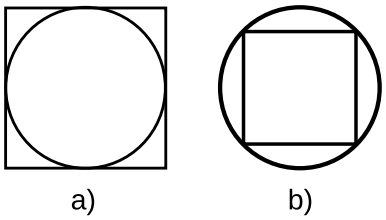


FIGURE 2: A) INSCRIBED CIRCLE IN A SQUARE. B) CIRCUMSCRIBED CIRCLE TO A SQUARE

A circumscribed recursion consists of a series of inscriptions and circumscriptions starting from a given initial circle. This circle can be inscribed in a square, and that square can have a circumscribed circle, and that circumscribed circle can be inscribed in another square... and so on.

The order of the recursion is given by the number of drawn figures. A recursion of order 1 is only the initial circle. A recursion of order 2 is the circle and the square in which the circle is inscribed. A recursion of order 5 will have 5 figures, alternating circles and squares as appropriate.

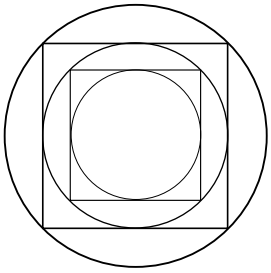


FIGURE 3: CIRCUMSCRIBED RECURSION OF ORDER 5. CONTAINS 5 FIGURES, 3 CIRCLES AND 2 SQUARES.

Given the radius of the center circle, can you calculate the area of the external figure? It can be a square or a circle.

INPUT

Input begins with a number T, the number of test cases ($1 \leq T \leq 9200$). Next, there are T lines, one for each test case. Each one of those lines contains 2 integer numbers R and K

($1 \leq R \leq 100, 1 \leq K \leq 92$), the radius of the initial circle (the most internal of the final figure) and the order of the circumscribed recursion.

OUTPUT

For each test case print a line with the area of the most external figure of the circumscribed recursion. If the most external figure is a square, print the area as an integer (round if necessary). If the most external figure is a circle, print the area in function of pi in the following way: "API" replacing A with the integer value that makes the answer valid (round if necessary). For example, the answer of the second test case is $2\pi = 6.283185...$, print only "2PI".

EXAMPLE

Input	Output
2 3 2 1 3	36 2PI

PROBLEM D. DREAMS

Source file name:	`dreams.c, dreams.cpp, dreams.java, dreams.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Colombia`

What happens when a dream does not become true?
Maybe it floats in the environment until it reaches another person
Or live in the subconscious of the frustrated dreamer.
Maybe it dies in the cemetery of dreams
Or it keeps waiting for a witty miracle.
What happens when a dream does not become true?
Ask to the dreams of the frustrated dreamer.

INPUT

First line of input contains an integer T, the number of test cases.

Each test case has two lines. The first line, contains two integers N and M ($1 \leq N \leq 1000$, $1 \leq M \leq N$), the total number of dreams of the frustrated dreamer, and the number of dreams that he has managed to fulfill. Consider that the dreams of the frustrated dreamer are listed from 1 to N. Next, there will be a line with M different integers D_i separated by a blank space ($1 \leq D_i \leq N$), representing the dreams that the frustrated dreamer has managed to fulfill.

OUTPUT

For each test case print a single line with the frustrated dreams of the frustratred dreamer in increasing order and separated by blank spaces. If the frustrated dreamer has already met all his dreams print a single blank line.

EXAMPLE

Input	Output
2 3 2 1 3 5 3 5 1 2	2 3 4

PROBLEM E. ELSEIF STATEMENT DOES NOT EXIST

Source file name: `elseif.c, elseif.cpp, elseif.java, elseif.py`
Input/Output: `standard`
Author('s): `Milton Jesús Vera - UFPS Colombia`

A group of students learned computer programming in the Microsoft VisualBasic programming language. Therefore, they believe that there are three flow control statements: if, else and elseif. The teacher has explained many times that the elseif statement does not exist, it is simply an if statement after an else statement using curly brackets, a block "elseif ... ". But the students still do not believe that to their teacher. In order to explain, the teacher wants his "elseif" students to help him convert ugly source code to pretty source code. Examples:

Case	Ugly Source Code	Pretty Source Code	Notes
-1	if(a > b) doSomething1(); else doSomething1();	doSomething1();	The if/else block is redundant. It is silly code valley! Do nothing.
0	if(a > b) doSomething1(); else doSomething2();	if(a > b) doSomething1(); else doSomething2();	It is not ugly source code. Do nothing.
1	if(a > b) doSomething1(); else ;	if(a > b) doSomething1();	Ignore and delete else statement.
2	if(a > b) ; else doSomething2();	if(a <= b) doSomething2();	Negate the condition and invert the code of if/else to reduce to case 1.

3	if(q == r){
	if(s != t) doSomething1();
	else doSomething2();
	}
	else doSomething2();
		if(q == r && s != t) doSomething1();
		else doSomething2();
			Put together with && (and) both conditions to reduce to only one block if/else.

4	if(i >= j) doSomething1();
	else{
	if(k <= l) doSomething1();
	else doSomething2();
	}
		if(i >= j || k <= l) doSomething1();
		else doSomething2();
			Put together with || (or) both conditions to reduce to only one block if/else.

5	if(u >= v) ;
	else{
	if(w == x) ;
	else{
	if(y != z) ;
	else doSomething2();
	}
	}
		if(u<v && w!=x && y==z) doSomething2();	Apply recursively case 2 and case 3 to reduce to case 1

For the ugly source code the following is always true:

- It is always syntactically correct.
- The condition of the ”if” is a simple relational expression with two operands and one operator (>, <, ==, !=, >=, <=).
- The operands are variables named with one lowercase letter (a,b,c,d...x,y,z).
- The only executable statements are the nothing statement ”;”, ”doSomething1();” and ”doSomething2();”
  - The nothing statement ”;” is always in the same line of an ”if” or ”else” statement.
  - The statement ”doSomething1();” is always only in the same line of an ”if” statement.
  - The statement ”doSomething2();” is always only in the same line of an ”else” statement.
- The statements ”;”, ”doSomething1();” and ”doSomething2();” do not use curly brackets ”{”, ”}”.
- Any ”if” statement always has an ”else” statement.
- The maximum levels of nested blocks ”else {if ...}” is four and the indentation is a blank space.
- The elseif statement does not exist, it is simply an if statement after an else statement using curly brackets, a block ”else {if ...}”.
- You can always apply any of the cases 0, 1, 2, 3, 4 and 5.

INPUT

Input contains multiple test cases. Each test case begins with a number n that indicates the number of lines of source code, then n lines of source code follows. The last test case is denoted by $n = 0$.

In all the cases the following rules are fulfilled:

- The condition of the if always carries a blank space before and after the operator.

- If the if does not have curly braces, there will always be a space after closing parentheses. If it does have curly braces, the opening curly brace goes right after closing parenthesis, without space.
- If the else has no curly braces, there will always be a space just after the word ”else”. If you have curly braces, no space is left.
- The indentation of a space will be used for each curly brace. That is, all the code between the most external curly braces, will start with a space on the left. If within these curly braces there are more curly braces, the code inside them will have two spaces to the left, and so on.

OUTPUT

For each test case print a number x, the number of lines of the pretty source code. Then print x lines containing the pretty source code. You should use the same format of spaces explained in the input.

EXAMPLE

Input	Output

2
if(a > b) doSomething1();
else doSomething2();
2
if(a > b)doSomething1();
else ;
2
if(a > b) ;
else doSomething2();
5
if(q == r){
if(s != t) doSomething1();
else doSomething2();
}
else doSomething2();
5
if(i >= j) doSomething1();
else{
if(k <= l) doSomething1();
else doSomething2();
}
8
if(u >= v) ;
else{
if(w == x) ;
else{
if(y != z) ;
else doSomething2();
}
}
0
	2
	if(a > b) doSomething1();
	else doSomething2();
	1
	if(a > b) doSomething1();
	1
	if(a <= b) doSomething2();
	2
	if(q == r && s != t) doSomething1();
	else doSomething2();
	2
	if(i >= j || k <= l) doSomething1();
	else doSomething2();
	1
	if(u < v && w != x && y == z) doSomething2();

Use fast I/O methods

PROBLEM F. FREE ROBOT

Source file name:	`robot.c, robot.cpp, robot.java, robot.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Colombia`

In the Robotics Programming Contest (RPC), the free robot has been the sensation: it moves in autonomous manner. The robot is placed in the upper left corner of a grid of dimensions $(N+1)*(N+1)$ and it moves N steps, being each step a move to any of its adjacent squares (horizontals, verticals or diagonals) without ever leaving the grid. The robot never goes through the same square twice. Knowing how many steps the robot will take, can you calculate how many different paths can the robot travel? The robot is always going to start in the same corner and two paths are considered different if at least one of their steps is different.

INPUT

The first line of the input contains a number T ($1 \leq T \leq 14$), the number of test cases. Each test case has a single line with an integer N ($1 \leq N \leq 14$), the number of steps that the robot will take.

OUTPUT

For every test case, print a line containing the number of different routes that the robot can travel in N steps.

EXAMPLE

Input	Output
2 1 2	3 15

PROBLEM G. ACROBAT

Source file name:	`acrobat.c, acrobat.cpp, acrobat.java, acrobat.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Colombia`

A circus has an acrobat who jumps on a surface such as the following:

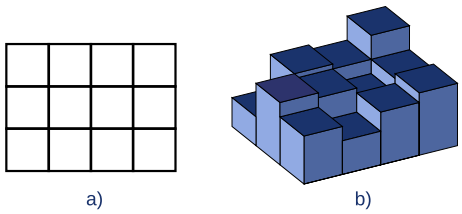


FIGURE 4: A) AERIAL VIEW OF THE SURFACE. B) SIDE VIEW IF THE SURFACE.

From the aerial view, we can represent the surface as a grid, but in reality each space of the grid is a box with base 1, but with different height (as can be seen in the side view). The acrobat is a jumps specialist and can move from one square to any of his 4 adjacent ones (not the diagonals, only the adjacent ones) with the following restrictions:

- If the acrobat wants to move to a cell with a higher box than the current, he can do it if and only if the difference of heights is not greater than 2 meters. His elasticity does not allow him to jump higher.
- If the acrobat wants to move to a cell with a smaller box than the current, he can do it as long as the difference of heights is less than 4 meters. A bigger jump could expose his health.

The acrobat always starts the performance in the lower left corner and ends in the upper right corner. He avoid his fatigue to the maximum and is conscious that climbing is more strenuous than descending. Therefore, he wants to go to the upper right corner, climbing the least possible amount of meters. If there are several possible paths, he will choose the one in which he has to descend the least possible amount of meters. Can you calculate what will be the number of meters that the acrobat will climb and descend until reach the upper right corner?

INPUT

Input starts with a number T, the number of test cases. Each case begins with two numbers l and w ($1 \leq l, w \leq 600$), the lenght and width of the surface in meters. Then, l lines come, each one with w integer numbers $h_{i,j}$ separated by a blank space, representing the height of each box in the surface ($0 \leq h_{i,j} \leq 12$).

OUTPUT

For each test case print a single line with two space-separated integers, the number of meters that the acrobat will climb and descend. If it is not possible to reach the upper right corner (there is no route that meets the previous restrictions) print ”IMPOSSIBLE” instead.

EXAMPLE

Input	Output
1 3 4 2 5 5 6 8 4 7 6 6 4 6 8	2 2

Use fast I/O methods

PROBLEM H. HARRY AND THE GOLDEN EGG

Source file name:	`harry.c, harry.cpp, harry.java, harry.py`
Input/Output:	`standard`
Author('s):	`Angie Melissa Delgado - UFPS Colombia`

Harry is in a very big trouble. The second task of the Triwizard Tournament is getting closer and he has not been able to decipher the message of the golden egg yet. Could you help him?

INPUT

Input begins with a number n, the number of test cases. Each case has multiple lines and contains the enigma that Harry has to decipher in order to complete the second task of the Tournament. Each enigma is formed by a set of w ($1 \leq w \leq 500$) words contaning lowercase and uppercase letters (only the first letter of a word can be uppercase) and the symbols single quote ('), point (.), comma (,) and semicolon (;). A single quote could be anywhere in the word, but a point, comma and semicolon will be always at the ends of the words. Each word has at most 100 characters. A line with an $*$ will mark the end of each test case.

OUTPUT

For each text case print the enigma deciphered. Leave an empty line between consecutive test cases.

EXAMPLE

Input	Output
1 Emoc kees su erehw ruo seciov dnuos, Ew tonnac gnis evoba eht dnuorg, Dna elihw uoy'er gnihcraes rednop siht; Ew'ev nekat tahw uoy'll yleros ssim, Na ruoh gnol uoy'll evah ot kool, Dna ot revocer tahw ew koot, Tub tsap na ruoh, eht tcepsorp's kcalb, Oot etal, ti's enog, ti now't emoc kcab. *	Come seek us where our voices sound, We cannot sing above the ground, And while you're searching ponder this; We've taken what you'll sorely miss, An hour long you'll have to look, And to recover what we took, But past an hour, the prospect's black, Too late, it's gone, it won't come back.

PROBLEM I. INCOMPLETE TEAM

Source file name:	`incomplete.c, incomplete.cpp, incomplete.java, incomplete.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Cúcuta`

Carlos, Crisel and Manuel have an excellent team of competitive programming. However, Manuel is also a recognized musician of the expansion group, with constant international tours and autograph signings, for which he can only attend each M competitions. Carlos, as well as being a programmer, is a chess player and represents his university in this sport. Unfortunately chess competitions almost always coincide with the programming ones, so he can only attend each N programming competitions. Crisel, on the other hand, is a consecrated programmer: she spends 100% of her time on programming competitions and none is lost.

Can you calculate which will be the first competition in which the whole team can participate?

INPUT

Input starts with an integer T, the number of test cases. Each test case contains two integers M and N ($1 \leq M, N \leq 10^4$).

Consider that the competitions are numbered in ascending order starting at 1, that there is an unlimited number of competitions, that the first competition that Jose Manuel will attend is the competition number M and the first one that Carlos will attend is the number N.

OUTPUT

For each test case, print a line with the number of the first competition that the entire team can attend.

EXAMPLE

Input	Output
4 2 3 4 4 2 4 3 7	6 4 4 21

PROBLEM J. SHUFFLE CARDS

Source file name:	`cards.c, cards.cpp, cards.java, cards.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Cúcuta`

The famous Magician Madler has shuffled his deck over and over, organizing it for his next trick. But it has suffered a shameful error: Some cards have been left face up, and others face down in the deck. The spectators are attentive, so he must solve the problem with style.

Madler places her deck of N cards on the table. Then take the M cards from the top of the deck and make an "inversion", that is, turn the entire set and then place them again on the remaining cards (He turns all the set of M cards, not one by one. In this way, the last of the M cards taken now will be at the top, while the one that was on the top will now be the last. In addition to change the order of the M cards, the turn will also modify their direction, those cards that were face up are now face down and vice versa).

Madler will keep repeating the same process with different values of M. In each occasion, M can be any value between 1 (only inverts the card that is in the top of the deck) and N (turn the entirely deck). Madler will stop when all the deck is facing down.

Obviously, Madler don't want to bore his spectators, and he will try to make the less possible number of inversions. Given the initial state of the deck, What is the less number of inversions that Madler has to do until all the deck is facing down?

INPUT

The first line of the input contains a number T ($1 \leq T \leq 1000$), the number of test cases.

Each case is represented with a line that contains a string S, the description of the deck. Each character of S represent a card, and the size of the deck is given by the size of S ($1 \leq |S| \leq 10000$). Each character of S can be 'a' if the card is facing up or 'b' if the card is facing down. The deck can be read from left to right so, the character in S_0 represents the top of the deck, S_1 the card that is under the top...S_{N-1} the last card of the deck that is just over the table.

OUTPUT

For each test case print a line with the less number of inversions that Madler has to do until all the deck is facing down.

EXAMPLE

Input	Output
5 ab ba aaba bbbbb aaaaa	1 2 3 0 1

PROBLEM K. UNIVERSAL LANGUAGE I

Source file name:	`universal.c, universal.cpp, universal.java, universal.py`
Input/Output:	`standard`
Author('s):	`Hugo Humberto Morales Peña - UTP Colombia`

In the holiday season at the beginning of the year Professor *Humbertov Moralov* is preparing some subjects for his course of *Automata Theory and Formal Languages*. He has given special attention to a procedure that enumerates all the words in the universal language (Σ^*) generated from an alphabet Σ:

Let $\Sigma = \{a, b\}$ be one sample alphabet, and let $k \in \mathbb{N}$ be the length of the words. For each k, there is a finite number of words over Σ with length k.

The words are lexicographically sorted. For convenience the empty string λ is enumerated as 0, then the words of length 1 are enumerated. In general the strings with length $k+1$ are enumerated after the ones with length k. Thus:

$$\lambda \ldots 0$$
$$a \ldots 1$$
$$b \ldots 2$$
$$aa \ldots 3$$
$$ab \ldots 4$$
$$ba \ldots 5$$
$$bb \ldots 6$$
$$aaa \ldots 7$$
$$\vdots$$

and so on.

Now the Professor Humbertov Moralov needs your help to determine the position of a word in the enumeration given by the procedure explained before over a group of strings written in an alphabet Σ.

INPUT

Input begins with an integer t ($1 \leq t \leq 10^3$), the number of test cases. The first line of the test case contains an integer m ($1 \leq m \leq 26$), the size of the alphabet Σ. The second line contains exactly m space-separated different symbols σ_1, σ_2, σ_3, ..., σ_m ($\sigma_i \in \{a, b, c, d, \ldots, z\}$, σ_i is any lowercase letter of the English alphabet). The alphabet Σ is sorted lexicographically. The test case continues with a line containing a number n ($1 \leq n \leq 10^3$), indicating the amount of words. Finally each of the following n lines contain a string w ($w \in \Sigma^*$, $1 \leq |w| \leq 10$, The size of the string is at most ten).

OUTPUT

For each test case, you should print n single lines, each one containing one positive integer p ($1 \leq p \leq 10^{15}$), denoting the position in the enumeration procedure for the word w on the alphabet Σ.

EXAMPLE

Input	Output
2 1 d 4 d dd ddd dddd 4 a m o r 6 amor roma amar aro rama mora	1 2 3 4 112 313 104 35 281 193

Use fast I/O methods

PROBLEM L. LÁZARO SYSTEM

Source file name:	`lazaro.c, lazaro.cpp, lazaro.java, lazaro.py`
Input/Output:	`standard`
Author('s):	`Gerson Lázaro - UFPS Colombia`

A fractal is a geometric object whose structure appears to be irregular, but in reality it is repeated at different scales. The Lázaro System is formal mechanism to denote fractals using a simple mathematical notation: Basic functions that indicates the actions to build the fractal and a general recursive function to define the object are created. The recursive function is a sequence of functions (basic or recursive) separated by commas that are evaluated one after another. At the end, a set of restrictions are defined. For example:

Basic functions

- *a()* = draw a stroke of length 1 in the current direction.
- *b()* = Turn $60°$ counterclockwise.
- *c()* = Turn $60°$ clockwise.

General recursive function

$$f(x) = \begin{cases} a() & if(x = 0) \\ f(x-1), b(), f(x-1), c(), c(), f(x-1), b(), f(x-1) & if(x > 0) \end{cases}$$

Restrictions

At the beginning the figure is in $0°$ (horizontal position) from origin.

With the previous example we can build figures like the following:

f(0) f(1) f(2)

The previous figure is a Koch snowflake, one of the best-known fractals. Although different fractals can be drawn under the Lazaro system, we will focus on this one. Given the function $f(n)$ can you draw the corresponding Koch snowflake?

INPUT

Input has multiple test cases. Each case consist of a line with an integer n ($0 \leq n \leq 7$).

OUTPUT

For each test case print the Kock snowflake corresponding to $f(n)$. Use _, / and \ to draw the figure. Each time that an instruction a() appears at $0°$ or $180°$, use _.

Each time that an instruction a() appears at $60°$ or $240°$, use $/$. Each time that an instruction a() appears at $120°$ or $300°$, use $\backslash$. Fill every step that is not a part of the Koch snowflake with '.'.

The draw con be considered as a grid of characters. Print a grid of sufficient size so that the curve is printed in full, but without printing any row or column containing only '.'. Print a blank line between each pair of test cases. Look at the example cases for clarity.

EXAMPLE

Input	Output
0 1 2 3	_ _/_ ...._/_.... ....\../.... _/_/.._/_ ...._/_............... ....\../............... ...._/_/.._/_........... ....\.........../........... ..../_........._\........... ....\...../............ ...._/_...._/....._...._/_.... ....\../....\......../....\../.... _/_/.._/_/........_/_/.._/_

PROBLEM M. VALYRIO MUÑO ENGOS ÑUHYS ISSA

Source file name:	`valyrio.c, valyrio.cpp, valyrio.java, valyrio.py`
Input/Output:	`standard`
Author('s):	`Angie Melissa Delgado - UFPS Colombia`

Valyrian is my mother tongue. David Peterson is a language creator and a recognized member of the Language Creation Society - LCS. As part of his work he has created many artificial languages for TV and Movies, including Dothraki and High Valyrian.

This year the LCS is looking for the most *beautiful* artificial language, based in a set of very complex rules. In order to minimize the judges work, the LCS defined two conditions that any artificial language must fullfill:

- The alphabet of the language only have vowels and consonants.
- All words are formed of letters. Letters can be a vowel or a consonant that belongs to the alphabet.
- A word is qualified as *beautiful* if any consonant in the word is immediately followed by a vowel.
- An artificial language must have at least k beautiful words of length l to be nominated as the *beautiful language of the year*.

David wants to know if any of his beloved creations fullfill the above conditions and could be nominated for the awards. Could you help him?

INPUT

Input begins with a number n, the number of test cases. Each case consists of one line with four integers c, v, l and k ($1 \leq c \leq 50, 1 \leq v \leq 50, 1 \leq l \leq 500, 1 \leq k \leq 10^9$), the number of consonants, the number of vowels, the length and the number of valid words that the artificial language must have.

OUTPUT

For each test case, print a single line containing the number of different valid words of length l modulo $10^9 + 7$, followed by the word $Accepted$ if the language fullfill the conditions or $Rejected$ otherwise.

EXAMPLE

Input	Output
3 1 1 4 4 1 2 2 10 4 3 2 20	5 Accepted 6 Rejected 21 Accepted