Practice Set
Problem A

Table

You are given two Strings s and t. All characters of s and t are distinct. No character of s is present in t and no character of t is present in s.

Let N be the length of s, and M be the length of t. Define a 2-dimensional string array "table" as follows:

- table[i][0] = s[i-1] (1<=i<=N)
- table[0][j] = t[j-1] (1<=j<=M)
- table[i][j] = min(table[i-1][j], table[i][j-1]) + max(table[i-1][j], table[i][j-1]) (1<=i<=N, 1<=j<=M)

Note that min and max are defined by the lexicographical order of strings, and A+B means the concatenation of strings A and B.

Your task is to find a substring of table[N][M]. Let L be the length of table[N][M]. Output the substring of table[N][M] whose start position (0-indexed) is pos and length is min(50, L-pos).

Input

First line of the input contains T the number of test cases. Each test case contains s, t and pos. pos will be between 0 and L-1. The length of s and t will be between 1 and 30 inclusive. s and t will contain only alphanumeric characters.

Output

For each scenario, output the substring of table[N][M] whose start position (0-indexed) is pos and length is min(50, L-pos).

Sample Input

```
5
ad cb 0
fox cat 0
M0HAXG COFU12 919
A B 0
da bc 0
```

Sample output

```
acbacd
acfcfoacftacfcfocfox
MOFU2
AB
abdbdc
```
Problem B

Rook

You will be given three integers N, M and K. You are to calculate the number of ways to place K rooks on a NxM chessboard in such a way that no rook is attacked by more than one other rook. A rook is attacked by another rook if they share a row or a column and there are no other rooks between them. To avoid problems with big numbers the calculation should be done modulo 1,000,001.

Input
First line of the input contains T the number of test cases. Each test case contains N (1≤N≤50), M (1≤M≤50) and K (1≤K≤100).

Output
For each test case output the number of ways to place K rooks on a NxM chessboard in such a way that no rook is attacked by more than one other rook.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>190</td>
</tr>
<tr>
<td>4 5 2</td>
<td>6</td>
</tr>
<tr>
<td>2 3 3</td>
<td>0</td>
</tr>
<tr>
<td>6 7 20</td>
<td>879507</td>
</tr>
<tr>
<td>50 25 50</td>
<td>340200</td>
</tr>
<tr>
<td>9 6 10</td>
<td>333990</td>
</tr>
<tr>
<td>41 42 23</td>
<td>878278</td>
</tr>
<tr>
<td>41 49 47</td>
<td>636557</td>
</tr>
<tr>
<td>49 49 59</td>
<td></td>
</tr>
</tbody>
</table>
Problem C

Common Subsequence

A subsequence of a string is obtained by removing zero or more characters from it. You are given three Strings, and must determine the number of different non-empty subsequences they all share.

Input
First line of the input contains \( T \) the number of test cases. Each test case consists of 3 strings (containing only lower case letters) separated by a single space. The length of each of these strings is between 1 and 60 inclusive.

Output
For each test case output the number of different non-empty subsequences they all share.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
</table>
| 4
call accelerate candle
no correct answer
abc bac cab
qppmmq qqrmpq pmmnqngtrn                   | 6             |
|                                              | 0             |
|                                              | 3             |
|                                              | 4             |
Problem D

Cycle Counting

You are given a directed graph $g$, and you must determine the number of distinct cycles in $g$ that have length less than $k$. Since this number can be really big, return the answer modulo $m$. A cycle is a non-empty sequence of nodes (not necessarily distinct) in which there is an edge from each node to the next node, and an edge from the last node in the sequence to the first node. Two cycles are distinct if their sequences are not identical.

Input
First line of the input contains $T$ the number of test cases. Next line contains 3 integers $k$ ($1 \leq k \leq 100000000$), $m$ ($1 \leq m \leq 1000000000$) and $n$ ($1 \leq n \leq 40$). $n$ is the number of nodes in the graph. Each of next $n$ line contains $n$ strings each of which contains exactly $n$ characters where the $j$th character of the $i$th line is ‘Y’ if there is an edge from node $i$ to $j$, and ‘N’ otherwise.

Output
For each test case output the number of distinct $k$ length cycles modulo $m$.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
</table>
| 5
6 100 4
NYNY
NNYN
YNNN
YNNN
1000000 100000000 4
NYNY
NNNN
YNYN
NYNN
1500 1 2
NY
YN
1000 10000 4
NYY
YNY
YYN
YNYN
796325 389526560 3
NNY
YNY
YYN | 12
0
0
3564
308364655 |
Problem E
Lights

You are given a 4x4 game board consisting of buttons which are either lit or unlit. Pressing a button changes the state of that button, along with the states of the buttons to the immediate left, right, top and bottom. Pressing and holding a button, which counts as two moves, changes only the state of that individual light. The goal of the game is to turn off all of the lights. And you need to do that with minimum number of moves.

Input
The first line of the input gives an integer T, which is the number of test cases. Each test case consists of a 4X4 grid of 0-1. 0 means the light is off and 1 means the light on.

Output
For each test case, output the least number of moves necessary to turn off all of the lights.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
</table>
| 4
| 1010
| 0111
| 1010
| 0110 |
| 0110
| 0000 |
| 1100
| 0001 |
| 0111 |
| 1010
| 1101 |
| 0110
| 1101 |
| 0110
| 1001 |
| 0110
| 0000 |
| 6
| 7 |
| 9 |
| 2 |
Problem F

The OR Game

Fox Ciel and Toastman are playing a game. The game uses a memory and N cards. Initially the value of the memory is set to zero. You are given an array cards containing exactly N integers. cards[i] is the number written on the i-th card. Ciel and Toastman take alternate turns, and Ciel plays first. In each turn, the player chooses a card and removes the card from the game (this card can't be used later). If the chosen card contains x and the value of the memory is y, the value of the memory is changed to (x | y). The '|' symbol stands for bitwise OR (see notes for clarification). If a player can't make a move (because there are no cards left), or if after a player's move the memory becomes 511, this player loses. Determine the winner when both players play optimally. If Fox Ciel wins, output "Fox Ciel" (quotes for clarity). If Toastman wins, output "Toastman" (quotes for clarity).

Input
The first line of the input gives an integer T, which is the number of test cases. Each test case starts with N(1≤N≤80). Next line contains N integers between 0 and 511 inclusive.

Output
For each test case, output the winning player name.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Fox Ciel</td>
</tr>
<tr>
<td>5</td>
<td>Fox Ciel</td>
</tr>
<tr>
<td>3 5 7 9 510</td>
<td>Toastman</td>
</tr>
<tr>
<td>4</td>
<td>Fox Ciel</td>
</tr>
<tr>
<td>511 12 2 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Fox Ciel</td>
</tr>
<tr>
<td>500 10 1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1 1 1 2 2 508</td>
<td></td>
</tr>
</tbody>
</table>
Problem G

Escape

You are a prisoner and you want to escape from your jail. The jail is a grid where each square is either free or impassible. There are one or more exits located in free squares. You are initially standing in a free square and your goal is to reach one of the exits. Unfortunately, your eyes are covered, so you cannot see where you’re going. Each time you move, you check up, down, left and right to see which of those four adjacent squares are free, and you randomly walk to one of the free squares. You continue to do this until you land on a square containing an exit. If there is no adjacent free cell, you stay at your current position.

Input

First line of the input contains T the number of test cases. Each test case starts with a line containing R and C denoting the number of rows and the number of columns in the grid. Each of the next R lines contains C symbols, where your start location is marked by the '@' character, impassible squares are marked with '#' , exits are marked with '$' and all other free squares are '.'.

Output

For each test case output the expected number of moves required to escape. Print 4 digits after the decimal. If it is impossible to escape, return -1.0000.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
</table>
| 4
1 2
@$
2 2
@#
#$
1 10
@........$ 8 7
..$....
.....#
..#..##$
.....#
..#..#
.#.....
....#@..
.###...
| 1.0000
1.0000
81.0000
130.8334 |
Problem H
Coloring in Hypercube

In an N-dimensional grid the co-ordinates of a cell are denoted as $X_1, X_2, \ldots, X_N$. Any cell with negative co-ordinate is colored white. The origin cell (the one with all zero co-ordinates) is colored as black. The color of a cell in $(X_1, X_2, \ldots, X_N)$ depends on the N cells with co-ordinates $(X_1-1, X_2, \ldots, X_N)$, $(X_1, X_2-1, \ldots, X_N)$, ..., $(X_1, X_2, \ldots, X_{N-1})$. The cell is colored white if and only if the number of black colored cells among these N co-ordinates is even; otherwise the cell is colored black.

You are given the starting and ending co-ordinate of sub-hypercube. You need to compute how many hyper cells in this sub hypercube are colored black.

**Input**
First line of the input contains $T(<51)$ the number of test cases. Each test case starts with a line containing $N$ the dimension of the hypercube. The second line contains $N(0<N<9)$ integers denoting the co-ordinate of the starting cell of the hypercube. The third line contains $N$ integers denoting the co-ordinate of the ending cell of the hypercube. All the co-ordinates will be non-negative integers with at most 15 digits.

**Output**
For each test case, output the number of black colored cells in the given hypercube. Since the result can be too big so output the result modulo $1000000009$.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>4 0 4</td>
<td>22</td>
</tr>
<tr>
<td>7 9 8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0 3</td>
<td></td>
</tr>
<tr>
<td>10 9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>0 3 0 2</td>
<td></td>
</tr>
<tr>
<td>6 8 1 5</td>
<td></td>
</tr>
</tbody>
</table>
Problem I

Queen Game

Queen game is played in a chessboard of size R rows and C columns. Rows are numbered from 1 to R and columns are numbered from 1 to C. The topmost square is in row 1 and column 1. The game is a 2 player game. Initially there are N queens placed in various squares of the chessboard. In his turn, the player picks a queen and moves it either towards the top vertically, or towards the left horizontally or towards the top-left diagonally. When the queen reaches the top-most square it is removed from the board. The player who gives the last move wins. Each square is big enough to accommodate infinite number of queens. The players give their moves by turns. You are given the size of the chessboard and the initial positions of the N queens. Assuming that both of the players play perfectly your task is to determine who will win this game.

Input

First line of the input contains T the number of test cases. Each test case starts with a line containing 3 integers R(1≤R≤10), C(1≤C≤10^{15}) and N(1≤N≤1000). Each of the next N lines contains the positions of N queens. The position is denoted by two integers. The first integer is the row number and the second integer is the column numbers.

Output

For each test case output “YES” if the first player has a winning strategy and “NO” otherwise.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
</table>
| 3
| 5 5 1
| 2 3
| 5 5 2
| 4 4
| 4 4
| 5 5 3
| 1 2
| 2 1
| 2 2 | NO
| NO
| YES |
Problem J

Taxi Cab Scheme

Running a taxi station is not all that simple. Apart from the obvious demand for a centralized coordination of the cabs in order to pick up the customers calling to get a cab as soon as possible, there is also a need to schedule all the taxi rides which have been booked in advance. Given a list of all booked taxi rides for the next day, you want to minimize the number of cabs needed to carry out all of the rides.

For the sake of simplicity, we model a city as a rectangular grid. An address in the city is denoted by two integers: the street and avenue number. The time needed to get from the address a, b to c, d by taxi is |a - c| + |b - d| minutes. A cab may carry out a booked ride if it is its first ride of the day, or if it can get to the source address of the new ride from its latest, at least one minute before the new ride’s scheduled departure. Note that some rides may end after midnight.

Input
On the first line of the input is a single positive integer N, telling the number of test scenarios to follow. Each scenario begins with a line containing an integer M, 0 < M < 500, being the number of booked taxi rides. The following M lines contain the rides. Each ride is described by a departure time on the format hh:mm (ranging from 00:00 to 23:59), two integers a b that are the coordinates of the source address and two integers c d that are the coordinates of the destination address. All coordinates are at least 0 and strictly smaller than 200. The booked rides in each scenario are sorted in order of increasing departure time.

Output
For each scenario, output one line containing the minimum number of cabs required to carry out all the booked taxi rides.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample Output</th>
</tr>
</thead>
</table>
| 2
| 2
| 08:00 10 11 9 16
| 08:07 9 16 10 11
| 2
| 08:00 10 11 9 16
| 08:06 9 16 10 11 | 1
| 2 |
Problem K

Homework

John has a list of integers first, and his task is to transform it so that it contains exactly the same elements as another list second (but not necessarily in the same order). To achieve this goal, John will perform a number of operations in sequence. These are the three operations that can be performed on a list:

- Add some arbitrary integers to the list. The number of elements added must not be more than the size of the list before the operation.
- Delete some elements from the list. The number of elements deleted must not be more than half of the size of the list before the operation.
- Arbitrarily change some elements in the list. The number of elements changed must not be more than half of the size of the list.

You are given the list first and second. Compute the minimum number of operations required to achieve John's goal.

Input

The first line of the input gives an integer T, which is the number of test cases. The first line of the test case contains N the number of elements in the list first. Next line contains N integers separated by a single space. The 3\textsuperscript{rd} line contains M the number of elements in the list second. The 4\textsuperscript{th} line contains M integers separated by a single space. N and M are between 1 and 50 inclusive. All other integers are between 0 and 1000 inclusive.

Output

For each test case, output the minimum number of operations required to achieve John's goal.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2 3 3</td>
<td>3</td>
</tr>
<tr>
<td>1 0</td>
<td>2</td>
</tr>
<tr>
<td>1 5 5 7 7</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>7 999 5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7 2 999 5</td>
<td>0</td>
</tr>
</tbody>
</table>