FINAL ROUND

June 15th, 2008

organized by

HIMTI (Himpunan Mahasiswa Teknik Informatika) BINA NUSANTARA

Problemset

A	Superstitious Skylab Tower
B	Panda Land 5: Panda Programming Language
C	Almost Clear
D	Eat or Be Eaten
E	Indomie
F	In Queries
G	Hotel
H	Walaweh

Instruction

You are given five hours to solve the problems in this contest. Input for each problem should be read from *standard input*. Outputs on the other hand should be printed to *standard output*.

Problem A

Superstitious Skylab Tower
Time Limit: 3s

The 21st century has brought exponentially-growing advances to structural engineering, such that by year 2100 there are already many orbital towers (a.k.a. space elevators) with heights reaching thousands of kilometers to the space.

In year 2100 a group of leading scientists from around the world plans to build another such tower as a research station, extending beyond the heights of the geostationary orbit. The floors in the tower will be numbered, starting sequentially from 0. However, many of the scientists were strongly superstitious and therefore they don't want to use floors numbered 4, 13, and any floor which number contains 4 and/or 13 as its substring.

So in order to overcome the problem and make effective use of all the available floors, the developer team has decided to skip such numbers in the floor numbering plan of the new tower. So the floor numbers will go something like this: ..., 3, 5, 6, ..., 12, 15, 16, ...

This decision solved the superstitions problem, yet caused another type of problem. Some of the scientists conduct ultra-high-precision Earth Science research projects and therefore need to know exact height of their observation spots (which can reach very high into the space), given the height of each floor (assumed to be exact and the same for all floors) and the floor number. Your task is to write a program calculating the heights of the floors given such parameters.

Input

The first line of input contains an integer T, the number of test cases follow.

Each case consists of two integers, k ($0 < k \leq 10^{12}$) the floors which height is requested, and h ($0 < h \leq$ 1,000,000) the height for each floor. You may safely assume that all numbers are valid, in other words, do not contain the numbers 4 and/or 13 as its substring which the scientists avoid.

Output

For each test case, print the heights of the floor on a single line

Sample Input	Output for Sample Input
5 1 3 5 3 15 3 6 888 12 888	3 12 36 4440 9768

Panda Land 5: Panda Programming Language
Time Limit: 5s

Unlike in human world, panda world has only one programming language, it is Panda Programming Language (PPL). This language is developed by Panda Bureau of Technology (PBT), one of the government's special department.

This language is built based on procedural programming's rule, so it's completely possible to create more than one function on a program. Unfortunately, this language lacks of function prototyping ability. That means each function should have been defined before it can called by any other function. Recursive call is allowed in PPL.

Currently PBT is too busy to build that feature. Meanwhile, there's a certain program that need to be compiled. The programmers wrote that program without taking the function definition sequence in consideration (ie. some functions are not defined before other function that call them). So, they came to human world and found you, the best programmer on earth! They need your help to write a specific program that can rearrange their program so it can be compiled. But they have one more problem, their programming environment are not so friendly as yours, so the total cost of rearrangement should be minimized!

The cost of moving a certain function is calculated by multiplying the number of line of that function to the total number of skipped line. For example, let there're four functions defined as A, B, C and D (A at the top and D at the bottom) which have 10, 4, 6 and 3 lines respectively. Moving A below D will cost A * (B + C + D) = 10 * (4 + 6 + 3) = 130. Moving D above B (between A and B) will cost D * (C + B) = 3 * (6 + 4) = 30.

Input
The first line of input contains an integer T, the number of test cases follow.

Each case begins with an integer N ($1 \le N \le 18$), the number of function. The next line contains N integers ($1 \le M_i \le 100$) representing the number of line for each function $i = 1 \dots N$. The next N lines each will describes ith function dependency. Each line will start with an integer C ($0 \le C < N$), the number of other function that is called by that ith function. Next will be C integers ($1 \le F_j \le N$) representing the function which it needs. The last line for each case will contains N integers representing be the initial function definition.

Output
For each case, print in a single line the minimum cost needed to rearrange that program so it can be compiled, or print "-1" if there's no such arrangement.

Sample Input	Output for Sample Input
2 4 7 3 12 8 1 3 1 4 2 1 3 0 1 2 3 4 5 7 3 12 8 4 3 1 2 4 0 1 2 0 1 4 1 2 3 4 5	-1 161

Problem C

Almost Clear
Time Limit: 5s

In ACM (Awkward Commodity Museum), like every other museum, collects many valuable objects from every place around the globe, therefore several surveillance cameras are installed to watch over the museum for possible unauthorized access. In order to assure maximum surveillance, Mr. Effendy as the Head Security of ACM proposed such layout that all valuable objects being displayed must be visible from one of the cameras.

In ACM, Mr. Halim from the QCD (Quality Control Department) is in charge to verify the layout Mr. Effendy proposed. Usually he develops program to help him do the verifying process, but he is currently busy helping some public forum to understand about Binary Search Tree and the likes of which a programmer like you should already mastered. You should help Mr. Halim before he's fired because stealing company's hour to do things irrelevant to his position.

Mr. Halim is asked to write a program that accepts a layout, and verify that all valuable objects are visible by at least one of the camera installed. You must help him! Mr. Halim is a very brilliant guy; he doesn't require you to write a complete solution. Write a program that receives 3 parameters: polygon A, polygon B, point C and determine whether polygon A (the valuable object) is "hidden" by polygon B (other object) when a camera is installed at point C. Note that your program is only need to deal with these 3 objects and nothing more, Mr. Halim will do the rest.

You may safely assume that:
1. Your program deals in 2-D coordinates.
2. Polygon A and B is convex.
3. The camera has infinite vision (meaning that it can see an object very far away).
4. The camera can rotate 360 degrees freely.
5. The camera **cannot** move.
6. All position will be valid. Polygons do not intersect each other; camera will not be inside any of the polygons.
7. All the characters and events described above are fictional. Any similarity of names of the characters and events is partially accidentally. No polygons were harmed during the making of this problem

Input
The first line of input contains an integer T ($1 \le T \le 1,000$), the number of test cases follow.

Each case begins with an integer M_1 ($1 \le M_1 \le 1,000$), the number of vertices of polygon A, and then followed by M_1 integer-pairs which are the X-coordinate and Y-coordinate of each vertex in counter-clockwise direction. The second line will be the description of polygon B in the same format and constraint as the description of polygon A. The third line will be 2 integers which are the X-coordinate and Y-coordinate of the camera respectively. All coordinates will be non-negative and fit in 31-bit integer.

Output
Output for each test case will be in one line. Print "CLEAR" if polygon A is not obstructed by polygon B at all, print "ALMOST CLEAR" if polygon A is obstructed partially by polygon B, or print "NO VISION" if polygon A is fully hidden by polygon B.

| Sample Input	Output for Sample Input
3 3 5 23 24 23 16 36 4 0 10 5 10 5 20 0 20 0 0 4 22 53 34 53 34 63 22 63 4 18 31 38 31 38 46 18 46 26 0 4 36 38 48 38 48 48 36 48 4 28 16 51 16 51 30 28 30 20 57	ALMOST CLEAR NO VISION CLEAR

Problem D

Eat or Be Eaten
Time Limit: 3s

Deep down under the sea, there're two kinds of living organism, let's say A and B. A is B's predator, but A will only eat B if and only if its size is strictly bigger than its prey. For example, let the size of A = {8, 1, 7, 3, 1} and B = {3, 6, 1}, then there are 7 pairs of A – B where A > B: 8 – 3, 8 – 6, 8 – 1, 7 – 3, 7 – 6, 7 – 1, 3 – 1.
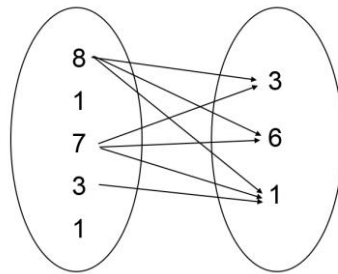
Given the size of each organism in A and B, write a program to count how many pair of A - B are there such that A is strictly bigger than B. Your program should be efficient as the number of organism may be large.

Input
The first line of input contains an integer *T*, the number of test cases follow.

Each case will begin with two integers *N* (1 ≤ *N* ≤ 20,000) and *M* (1 ≤ *M* ≤ 20,000), denoting the number of organism A and B respectively. The next line will contain *N* positive integers represent the size of each A organism. The third line will contain *M* positive integers represent the size of each B organism.

Output
For each case, print on a single line the number of pair A – B such that A is strictly larger than B.

Sample Input	Output for Sample Input
2 5 3 8 1 7 3 1 3 6 1 3 4 2 13 7 103 11 290 215	7 1

Problem E

Indomie
Time Limit: 3s

During recession, Felix needs to queue for SembakoPlus. Sembako, as we all know, stands for "Sembilan Bahan Pokok" which consists of 9 kinds of item: Rice, Sugar, Cooking-oil, Meat, Egg, Milk, Corn, Kerosene and Iodized Salt. SembakoPlus consists of Sembako and one more item: Indomie! Felix's favorite of all time!! (therefore, no wonder why he could stand for this long queue)

Each person in the queue is allowed to pick only one item. No need to ask, Felix wants only Indomie. Unfortunately, they are running out of SembakoPlus stock and currently there are three kinds of item left: Rice, Sugar and Indomie. As he could see from afar, he is quite sure that Rice and Sugar will be enough for everybody.

Given the number of remaining Indomie and the number of people queuing in front of Felix, your task is to count the probability that he will get his Indomie. Felix can't do programming right now as he is very nervous so he can't think logically. He needs your help!

Input
There will be multiple test cases for this problem. Each test case contains two integers N ($1 \leq N \leq 50$) and S ($0 \leq S \leq 50$), where N is the number of people queuing in front of Felix and S is the remaining number of Indomie.

Output
For each case, print in a single line the probability in percentage that he will get his Indomie with 5 digits precision (he's being paranoid)

Sample Input	Output for Sample Input
2 1	50.00000
3 2	76.92308
4 0	0.00000
4 1	33.33333
10 10	99.99831
14 9	98.65515
30 14	95.16071

Explanation for 1st sample test case:
There are two peoples queuing in front of Felix, so those two peoples could pick of the following combination {1st people, 2nd people}:
1. Rice, Rice
2. Rice, Sugar
3. Rice, Indomie
4. Sugar, Rice
5. Sugar, Sugar
6. Sugar, Indomie
7. Indomie, Rice
8. Indomie, Sugar

Since there is only one Indomie left, there are only 4 out of 8 combinations that ensure Felix to get his Indomie (1, 2, 4 and 5), hence the probability is 4/8 = 50%.

In Queries
Time Limit: 3s

Currenty ITComSysSC (Information Technology Computer System Student Club) needs to develop a software module which able to handle some query on data, and here they come to ask you. All the data will be integer and stored in a table of 5 columns. For the development purpose, you will use the initial data which are generated by this function:

$f(r, c) = (a \cdot f(r - 1, c) + b \cdot f(r, c - 1) + x) \bmod m$
$f(r, c) = 0$, for all r = 0 or c = 0.
$f(r, c)$ = the value on row r and column c, where $r = 1 \ldots n$, and $c = 1 \ldots 5$.

a, b, x, m and n are seeds for the function and will be given as input.

Here are the queries that you should handle:

Query	To Do	Example
insert *a b c d e*	append $\{a, b, c, d, e\}$ to the last row. a = data for 1st column, b = data for 2nd column, …, e = data for 5th column $(0 \le a, b, c, d, e < m)$.	insert 2 10 3 4 7
remove *r*	delete row r.	remove 3
max *c*	output the highest value in column c.	max 3
min *c*	output the lowest value in column c.	min 1
range *c a b*	output the number of row which value in column c between a and b (inclusive).	range 2 1 10

Each time a remove command is executed, the specified row will be deleted but all other rows below it will not be shifted (which means the row still there, only the data is emptied). If the row to be removed is already deleted/empty or out of range, then do nothing. Insert command will always append the data to the last row, even if there're empty rows before it (deleted rows).

You may safely assume that there's no output query (max/min/range) on empty table.

Input
The first line of input contains an integer T, the number of test cases follow.

Each case begins with five non-negative integers a, b, x, m $(1 \le m \le 10,000)$ and n $(0 \le n \le 100,000)$ the seed for generating the initial data. The next row contains a single integer q $(0 \le q \le 100)$ denoting the number of queries. The next q lines each represents the query in one of the above formats.

Output
Print "Case #*X*:" (*X* is the case number) at the first line of each test case. The following lines will be the output of the test case (each output on a single line).

Sample Input	Output for Sample Input
3 1 1 1 5 6 9 max 5 remove 2 remove 4 min 2 insert 2 4 6 3 10 range 4 1 5 insert 0 0 0 0 0 remove 7 max 3 3 3 3 3 3 2 remove 1 insert 1 1 1 1 1 2 4 3 10 2 6 remove 1 remove 2 insert 2 2 2 2 2 remove 1 remove 10 max 3	Case #1: 1 0 4 4 Case #2: Case #3: 2

Hotel

Time Limit: 3s

Indonesia tourism board has just sent a list of available room type of all hotels in Jakarta for teams that will participate in Indonesia Programming Contest 2008. Each record on the list contains:
1. Hotel name (max. 25 char, alphabet only).
2. Bed size (20 – 62).
3. Room capacity (1 – 4).
4. Number of available room (1 – 50).
5. Cost per room (1 – 5,000).

To simplify the problem, let's assume that each hotel will offer only one type of room (which means they will appear only once in the list).

Several participants have submitted their hotel preference to the committee, which consists of:
- Prefered bed size, grouped into three categories:
  o Type A: bed size 20 – 35
  o Type B: bed size 36 – 48
  o Type C: bed size 49 - 62
- Number of people in their teams (1 – 200).
- Maximum number of person in a room (1 – 4). The number of people in each room will be limited to this number even if the room has more capacity.

Based on the data above, write a program to find the cheapest hotel for each team. If there're more than one cheapest hotels, then choose one with largest bed size. If there're still more than one, then choose one which come first on the list.

You don't have to worry about multiple teams assigned at one hotel. What we will do here is only make a suggestion for each team, not a reservation.

Input

The first line of input contains an integer T, the number of test cases follow.

Each case will begin with two integers N ($1 \le N \le 50$) and M ($1 \le M \le 50$) the number of available hotel and the number of teams respectively. The next N lines each will contains four integers (bed size, room capacity, number of available room and the price per room) and a string which denotes the hotel's name. The next M lines each will contains three data: bed size type (A, B or C), number of people in their teams and maximum number of person in a room.

Output

Print "Case #X:" (X is the case number) at the first line of each test case. For each team, print on a single line the total cost and the hotel name which you suggested (in the same order as the team appearance in the input), separated by a single space. If there're no hotels that match the team's criteria, then output "no-hotel" (without quotes).

Sample Input	Output for Sample Input
2 2 3 40 3 2 10 MyHotel 37 4 5 50 HisHotel B 5 3 A 3 4 B 7 2 4 2 30 2 5 10 IndigoHotel 35 2 5 10 PurpleHotel 36 2 5 10 GreenHotel 36 2 5 10 BrownHotel A 6 2 B 6 2	Case #1: 20 MyHotel no-hotel 200 HisHotel Case #2: 30 PurpleHotel 30 GreenHotel

Problem H

Walaweh
Time Limit: 8s

Walaweh number is a numbering sequence that is so troublesome (that's exactly where it gets its name, "Walaweh!"). Walaweh number is similar to binary number (only consist of zeros and ones) except that the length of the number is important (thus leading zeros are preserved). Note that the "length" of Walaweh numbers means the number of digits in the Walaweh numbers.

To simplify the wording, Walaweh numbers of length L will be written as W_L, which denotes all Walaweh numbers with exactly L digits. Walaweh numbers (of any length) is an ordered list of numbers. The most basic (smallest) Walaweh numbers is W_1 which are "0" and "1" in that order. W_L can be generated from W_{L-1} except for W_1 which is fixed. This is done by creating two clones (C1 and C2) of W_{L-1} then apply some operations (see below) on C1 and C2 to produce C1' and C2'. The combined list of numbers in C1' followed by the list of numbers in C2' (in that order) produces W_L.

These are the 8 possible operations on C1 and C2:
1. Append a digit zero to the end of all numbers in C1 and append a digit one to the end of all numbers in C2.
2. Append a digit zero to the beginning of all numbers in C1 and append a digit one to the beginning of all numbers in C2.
3. Append a digit one to the end of all numbers in C1 and append a digit zero to the end of all numbers in C2.
4. Append a digit one to the beginning of all numbers in C1 and append a digit zero to the beginning of all numbers in C2.
5. Reverse the order of the list of numbers in C2 and do operation 1 above.
6. Reverse the order of the list of numbers in C2 and do operation 2 above.
7. Reverse the order of the list of numbers in C2 and do operation 3 above.
8. Reverse the order of the list of numbers in C2 and do operation 4 above.

W_1 is fixed. W_2 is generated by applying the first operation on W_1. W_3 is generated by applying the second operation on W_2 and so on... and it will go back to the first operation again after the eighth operation. So, W_9 is generated by applying the eighth operation on W_8. W_{10} is generated by applying the first operation on W_9 and so on... Walaweh!

Below is the list of W_1, W_2, W_3, and W_4 :

Walaweh Length	Sequence Number	Walaweh Number		Walaweh Length	Sequence Number	Walaweh Number
1	1	0		4	1	0001
1	2	1		4	2	0101
				4	3	0011
2	1	00		4	4	0111
2	2	10		4	5	1001
2	3	01		4	6	1101
2	4	11		4	7	1011
				4	8	1111
3	1	000		4	9	0000
3	2	010		4	10	0100
3	3	001		4	11	0010
3	4	011		4	12	0110
3	5	100		4	13	1000
3	6	110		4	14	1100
3	7	101		4	15	1010
3	8	111		4	16	1110

To give you an idea of "reverse the order of the list of numbers in C2" for the fifth to eighth operations, we give the last 5 numbers of W_6 :

Walaweh Length	Sequence Number	Walaweh Number
6	60	110011
6	61	101111
6	62	100111
6	63	101011
6	64	100011

Your job is to convert from Walaweh Length + Sequence Number into Walaweh Number and vice versa.

Input

There are multiple input, each on a line by itself. The line will either begin with the word "Walaweh" then followed by an integer number $L < 64$ and $N < 2^L$ or begin with the word "Sequence" then followed by a binary representation of the Walaweh number with length < 64.

Output

For input line that begins with "Walaweh" you have to output the N'th Walaweh number of length L. For those lines that begins with "Sequence" you have to output the Sequence number of the given Walaweh number (The length of the Walaweh number is already obvious from the input).

Sample Input	Output for Sample Input
Walaweh 1 1 Walaweh 3 6 Walaweh 4 13 Sequence 1100 Walaweh 5 14 Sequence 1110 Sequence 01010 Walaweh 6 1 Walaweh 6 20 Walaweh 6 32	0 110 1000 14 11100 16 31 100010 001110 011100