HKUST Local Contest Fall 2009
September 20, 2009

Contest Time: 1:00pm - 5:30pm

Letter	Page	Time limit	Memory limit	Name
A	2	2 sec	64MB	High Scores
B	3	10 sec	64MB	Randomized Sorting Algorithm
C	4	5 sec	64MB	Yet another simple problem
D	5	2 sec	64MB	The Return of the Swaps
E	6	10 sec	64MB	Addicted to Addition
F	7	60 sec	64MB	Cheapest Land

Contest Organizer:
Prof. Ke Yi
Mr. Derek Hao Hu

Problemsetter:
Mr. Derek Hao Hu

Judges:
Mr. Derek Hao Hu
Mr. Ji Luo
Mr. Tong Zhu

Special thanks to Mr. Tong Zhu for helping verify these problems.

Contest Rules and Regulations:

1. **This contest is an individual contest. Discussions between contestants are strictly prohibited.** Sanctions will be imposed on contestants if they are found to have violated the regulations governing integrity and honesty.
2. In this contest, **the contestants are given six programming problems**. The goal is to solve as many problems as possible. For those who solve the same number of problems, the one with lower score wins. (The scoring system will be explained below.)
3. **The programming languages to be used in this contest are C/C++ and JAVA.** The contestants use PC^2 to submit their source codes to the judge and the source codes are compiled by Visual Studio C++ 6.0, Visual Studio C++ 2005 or JAVA.
4. **The contestant should read the input and write the output via standard I/O.** The contestants can assume that all test cases are of the format as stated in the problem statements. i.e. No exception handling is needed.
5. The correctness of each submission is judged by inputting test cases into the submitted program. The submission is regarded as correct if its outputs match completely with the model outputs. The submission is judged as correct or wrong. **No partial credit is given.**
6. The contestants can re-submit another source code after previous wrong submissions.
7. **All programs should not run for more than the time limit specified in the problem** (in most cases a "correct" implementation will run far less than the time limit we provide).
8. **The contestants are ranked firstly by the number of problems solved, and secondly the total time spent on solving the problems.** Time spent on solving one problem is the time between the start of contest and the submission of the correct implementation of that problem. For each problem you solved, a penalty of 20 minutes will be added to your score for each wrong submission of that problem.
9. **The contestants are allowed to bring any hard copies of books, notes, references, dictionaries and sketch papers to the contest site.** Electronic devices are forbidden.

Problem A. High Scores

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Derek enjoys playing a game called "Plants vs. Zombies". (We guarantee that you won't have extra difficulty solving this problem if you haven't played this game before.) Just like many other games, this game contains a high score list, where the highest achieved scores are sorted in non-ascending order. If several scores are equal, their rank is the smallest position of such a score in the ranked list. For example, if the high score list is 100, 90, 90, 80. Then the rank list would be 1, 2, 2, 4 instead of 1, 2, 3, 4.

Now Derek has finished a round of game again and he found that his score haven't appeared on the high score list but he thinks it SHOULD APPEAR! So now here's your problem. You are given the number of possible entries in the high score list, a list of scores which are already in the high score list and a new score Derek has just acquired.

You should write a program to calculate the rank of the new score within the high score list. If the score is too low, then output -1. In a case where all places in the high score list are already filled, an old score will only be replaced if the new score is better.

Input

The first line of the input is an integer T, which indicates the number of test cases.
T test cases follow. Each test case starts with three integers: M, N and P, where M indicates how many scores already exist in the high score list, N indicates the new score Derek has just accomplished and P is the number of possible entries in the high score list ($10 \leq P \leq 50, 1 \leq M \leq P, 0 \leq N \leq 2,000,000,000$). A line with M integers follow, which are the values of the current scores in the high score list. The value of each score is between 0 and 2,000,000,000 and the M integers are sorted in non-ascending order.

Output

You should output T lines exactly. Each line consists of an integer, which is the rank of the new score N. If the score cannot get a position in the high score list, output -1 instead.

Example

Standard Input	Standard Output
3 3 90 10 100 90 80 10 1 10 10 9 8 7 6 5 4 3 2 1 10 1 10 10 9 8 7 6 5 4 3 3 0	2 -1 10

Problem B. Randomized Sorting Algorithm

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	10 seconds
Memory limit:	64 megabytes

Sorting is one of the most fundamental problems in computer science and I think all of you are familiar with many sorting algorithms like Bubble Sort, Insertion Sort, Merge Sort, Radix Sort, Counting Sort, Heap Sort, Quick Sort, ... Well, there just are so many sorting algorithms! But Derek still believes he can discover some new sorting algorithms that no one has ever discovered before. Now, your task is to help analyze one algorithm Derek has just thought out.

You are given an array A which contains a permutation of the first n positive integers and you need to sort them in ascending order. Derek told you that sorting can be done in a series of swaps and you only need to handle the inversions. An inversion is a pair (i, j) such that $i < j$ and $A[i] > A[j]$. The basic idea of Derek's algorithm is as follows: among all these inversion pairs, you choose one randomly and swap $A[i]$ and $A[j]$. Each pair has the same probability of being chosen. Now you need to calculate the expected number of swaps you need in order to sort the array A in ascending order.

If we reach a nice conclusion, we can publish a SODA paper together! Therefore, please try your best to solve this problem!

Input

The first line of the input is an integer T, which indicates the number of test cases.
T test cases follow. Each test case starts an integer $N(1 \leq N \leq 8)$ indicating the number of elements in the permutation. Then N integers follow, each of which is an integer between 1 and N and each number would appear exactly once.

Output

You should output T lines exactly. Each line consists of a value which is the expected number of swaps you need in order to sort the array A. The answer is rounded to the nearest 5 digits after the decimal point (0.00001).

Example

Standard Input	Standard Output
4 3 1 3 2 4 4 3 2 1 1 1 6 2 5 1 6 3 4	1.00000 4.06667 0.00000 5.66667

Problem C. Yet another simple problem

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	5 seconds
Memory limit:	64 megabytes

This problem is really short and simple.

Given n and k, calculate:

$$(1^k + 2^k + 3^k + \ldots + n^k) \bmod 1000000007$$

Input

The first line of the input is an integer T, which indicates the number of test cases.
T lines follow, each line contains only two integers n and k. $(1 \leq n \leq 10^9, 1 \leq k \leq 50)$

Output

You should output T lines exactly. Each line corresponds to the value of the formula with the corresponding n and k.

Example

Standard Input	Standard Output
3 5 1 4 2 13 5	15 30 1002001

Problem D. The Return of the Swaps

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	2 seconds
Memory limit:	64 megabytes

You are given an integer N, and you are allowed to perform the following operation: take two non-zero digits of N, decrease each of them by one and then swap the resulting digits. For example, if N is 166, you can reach the following numbers in one operation: 506 (swap '1' and the first '6'), 155 (swap the '6's) and 560 (swap '1' and the last '6').

You can do these swap operations as many times as you want. But can you tell me what's the largest number you can reach?

Input

The first line of the input contains an integer T, indicating the number of test cases. T lines follow, each line contains only one integer $N(1 \leq N \leq 100,000)$.

Output

You should output T lines where each line indicates the largest number you can reach using the swap operation.

Example

Standard Input	Standard Output
4 166 3499 34199 80970	560 8832 88220 80970

Problem E. Addicted to Addition

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Have you ever been punished in your primary school? Derek has been the favorite of teachers when it comes to punishment. Usually he spends night at home writing the same things again and again (like some difficult Chinese characters, some mathematics formulas), this time the task is even more tedious.

Simply because Derek gave a wrong answer of an addition in class, he is forced to add and add and add and add, finally addicted to addition. He is given a piece of paper with an integer A written on it, then an integer X is repeatedly added to the number on paper until the original number becomes B.

At any time, only one number is written on the paper. Therefore, we need to erase old numbers and the teacher even does not afford to give Derek new rubbers! Therefore, to save the cost of buying rubbers, only digits that need changing are erased and replaced by new ones. If there are new digits, they can only be added to the left of the number already written.

The shop will close very soon so Derek needs a program to help calculate how many erasers he will need. Therefore, please help him by calculating the total number of digits erased if the additions are performed above! The shop will really be closed very soon so therefore he needs a fast algorithm as well.

Input

The first line of the input contains an integer T, indicating the number of test cases. T lines follow. Each line has three integers A, B and $X (1 \leq A, B, X \leq 10^{16}, A < B)$. $B - A$ will be divisible by X.

Output

You should output T lines where each line indicates the number of digits erased to perform the required additions.

Example

Standard Input	Standard Output
6 2 6 1 88 107 19 123 843 120 3 811 404 2043601634821768 4274721675435952 3653414 12 123 111	4 2 12 4 4522764117 2

Problem F. Cheapest Land

Input file:	`Standard Input`
Output file:	`Standard Output`
Time limit:	60 seconds
Memory limit:	64 megabytes

Long long ago, there exists a rectangular grid. Sorry, should be "you have a rectangular grid". Yes, you have a rectangular grid, and each cell in this rectangular grid contains an integer number indicating the cost of this cell. (The cost is not necessarily positive). You are required to find the cheapest connected set of cells in this rectangular grid (this set can be empty, in which case the corresponding cost is 0).

The cost of the set of cells is the sum of the costs of its sells. Two cells are "adjacent" if they share a side (therefore a non-border cell has four adjacent cells). A set is connected if you can get from any cell in the set to any other cell in the set by moving between adjacent cells in this set.

Input

The first line of the input contains an integer T, indicating the number of test cases. T test cases follow, each test case starts with N and $M (1 \leq N, M \leq 9)$ indicating the numbers of rows and columns of the rectangular grid. Then we have N lines in each test case, which describe the costs of each cell in the rectangular grid.

Output

You should output T lines where each line indicates the cost of the cheapest connected set of cells you can select.

Example

Standard Input	Standard Output
2 2 2 -10 1 2 -10 3 3 1 2 3 4 5 6 7 8 9	-19 0