Nivel “Paradigmas”
Problem Set

13 de Octubre de 2017

Problem 1: Atoms in the Lab

SPOJ code: ATOMS

Mr. Yagami is a scientist in the Bhabha Atomic Research Centre. They are conducting a lab experiment on nuclear fission. In nuclear fission, one atom breaks into more than one atom of the same type.

Initially, there are **N** atoms in the lab. Starting from now (**t**=0), after each second, every atom will break into **K** atoms of the same type. They don't want the number of atoms to exceed **M**, so they have to stop the reaction at some time **t**=**T**. Can you find this value **T** for Mr. Yagami.

Input Format

First line contains **P**, the number of test cases. Next **P** lines contain three integers each. These three integers represent the values of **N**, **K** and **M** respectively.

Output Format

For each test case print the time at which the reaction will have to be stopped.

Constraints

1 ≤ **P** ≤ 10^4

2 ≤ **N**, **K**, **M** ≤ 10^18

Example

Input	Output
2 2 2 7 2 2 8	1 2

Explanation

1st test case:
at t=1, number of atoms=4
at t=2, number of atoms will be 8.
So reaction has to be stopped at t=1.

2nd test case:
at t=1, number of atoms=4
at t=2, number of atoms will be 8.
at t=3, number of atoms will be 16.

Problem 2: Amusing numbers

SPOJ code: TSHOW1

Amusing numbers are numbers consisting only of digits 5 and 6. Given an integer **k**, display the **k**-th amusing number.

Input

First line consists of integer **N** representing number of test cases.

Next **N** lines consist of **N** integers (1 <= **k** <= 10^15)

Output

N lines each displaying corresponding **k**-th amusing number

Example

Input	Output
2 1 5	5 65

Problem 3: RK Sorting

SPOJ code: RKS

RK is a great code breaker. He knows any cipher in the world can be broken by frequency analysis. He intercepted an enemy message. The message consists of **N** numbers, smaller than or equal to **C**. RK belives frequency analysis consists of sorting this sequence so that more frequent numbers appear before less frequent ones.

Formally, the sequence must be sorted so that given any two numbers X and Y, X appears before Y if the number of times X appears in the original sequence is larger than the number of time Y does. If the number of appearances is equal, the number whoes value appears sooner in the input should appear sooner in the sorted sequence.

Help RK by creating a "frequency sorter".

Input

First line of input contains two integers, **N** (1 <= **N** <= 1000), the length of the message, and **C** (1 <= **C** <= 10^9), the number from task description. Next line contains **N** integers smaller than or equal to **C**, which are the message itself.

Output

First and only line of output should contain **N** numbers, the sorted sequence.

Examples

Input	Output
9 3 1 3 3 3 2 2 2 1 1	1 1 1 3 3 3 2 2 2

Input	Output
5 2 2 1 2 1 2	2 2 2 1 1

Problem 4: How to Handle the Fans

SPOJ code: AKVQLD03

Trey Parker and Matt Stone, the creators of "South Park" are having some problems handling their fans. The number of fans is so huge that they can't even count them properly. So they hired "N" employees for counting the fans. All the "N" employees had their own separate offices and they were located in a straight line with positions numbered as 1, 2, 3 … up to N. Fans can come to the office of any employee at any time and tell them how they feel about the show and if they are lucky enough, they may get to meet Trey Parker and Matt Stone.

All the employees keep on updating Trey and Matt about the number of fans currently in their offices, so at each moment, they will have a list of "N" positions and the number of fans in each of these positions. Trey and Matt suddenly start taking a walk from office at position "A" to position "B" to meet their fans, but before they start walking they want to know the sum of all the fans in the offices from position "A" to "B". But counting them one by one is taking a lot of time, so now they hired you, an awesome software engineer to do this task. Your task is to find the sum of all the fans present in the offices between positions "A" to "B" ("A" and "B" inclusive). Let's see if you could do it fast enough.

Input

The first line of Input contains two integers "N" and "Q". "N" is the no. of employees hired by Trey and Matt. "Q" is the no. of queries to be followed.

Each of the next "Q" lines contain a query. A query can be of two types:

- "add P F" – this means that "F" no. of fans came to the office at Position "P"
- "find A B" – this means that Trey and Matt wants to know the sum of fans present at offices at positions "A" to "B"

Output

For each query of the type "find A B", output the sum of fans present at offices at positions "A" to "B" in a different line.

Constraints

1 <= N <= 10^6

1 <= Q <= 10^5

1 <= A < B <= N

1 <= P <=N

1 <= F <= 10^4

Example

Input	Output
10 10 find 1 5 add 5 8 add 6 2 find 4 5 find 4 6 add 2 4 find 2 6 add 6 7 find 1 6 find 7 10	0 8 10 14 21 0

Problem 5: Easy Problem

SPOJ code: EASYPIE

Last year there were a lot of complaints concerning the set of problems. Most contestants considered our problems to be too hard to solve. One reason for this is that the team members responsible for the problems are not able to evaluate properly whether a particular problem is easy or hard to solve. (We have created until now so many problems, that all seems quite easy.) Because we want our future contests to be better we would like to be able to evaluate the hardness of our problems after the contest using a history of submissions.

There are a few statistics that we can use for evaluating the hardness of a particular problem: the number of accepted solutions of the problem, the average number of submissions of the problem and the average time consumed to solve it (as "General rules" of the contest state "the time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the accepted run"). For the latter two statistics we consider only the teams which solved this particular problem. Needless to say we ask you to write a program that computes aforementioned statistics for all problems.

Task

Write a program that:

- reads a history of submissions during an ACM contest,
- computes for each problem the number of accepted solutions of the problem, the average number of submissions and the average time consumed to solve it,
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of the input contains one integer n (1 <= n <= 2000) being the number of submissions during the contest. Each of the next n lines describes one submission and contains a submission time (measured in seconds from the beginning of the contest), a team identifier, a problem identifier and a result of evaluating the submission separated by single spaces. The submission time is a positive integer not greater then 18000. The team identifier is a non-empty string consisting of at most five small letters or digits. The problem identifier is a capital letter A, B, ..., or I. The result is a capital letter A (the submission is accepted) or R (the submission is rejected).

Submissions are given in nondecreasing order according to submission times and there are 62 teams competing.

Please note that if a problem is accepted all further submission of this problem by the same team are possible but they should not be taken to the statistics.

Output

For each test case the output consists of nine lines. The first line corresponds to problem A, the second line to problem B, and so on. Each line should contain the problem identifier, the number of accepted solutions of the problem, the average number of submissions done by teams that solved that problem and the average time consumed to solve it separated by single spaces. The latter two statistics should be printed only if there was at least one accepted solution of the given problem and should be rounded to two fractional digits (in particular 1.235 should be rounded to 1.24).

Example

Sample Input	Sample Output
1 12 10 wawu1 B R 100 chau1 A A 2000 uwr2 B A 2010 wawu1 A R 2020 wawu1 A A 2020 wawu1 B A 4000 wawu2 C R 6000 chau1 A R 7000 chau1 A A 8000 pp1 A A 8000 zil2 B R 9000 zil2 B A	A 3 1.33 3373.33 B 3 1.67 4340.00 C 0 D 0 E 0 F 0 G 0 H 0 I 0

Problem 6: Encode Integer

SPOJ code: SNGINT

Given an integer N (0 <= N < 10^7). Encode N into another possible smallest integer M (M > 0), such that product of digits of M equals to N.

Input

First line input is T (total no. of test cases), followed by T (T < 10001) lines containing integer N.

Output

For each N output M or -1 (if encoding is not possible) in each line.

Example

Input	Output
3 24 5 11	38 5 -1

Problem 7: Queue (Pro)

SPOJ code: QUE2

There are N people standing in a Queue. You are given the height of each person and the number of people who are taller and standing ahead of him. You have to find the position of each person.

Input

First line contains a single integer T, the number of test cases. It is followed by T test cases each of which contains 3 lines. First line of each test case contains a single integer N. Second line contains N integers representing the heights of these N people. Third line also contains N integers denoting the number of taller people standing ahead of him.

Output

Output one line for each test case which contains the heights of the N people in the order in which they are standing.

Constraints

0 < T <= 20

0 < N <= 10000

Expected Time Complexity = O(N log N)

Example

Sample Input	Sample Output
1 5 33 11 22 44 55 0 2 1 1 0	33 22 11 55 44