Problem A. Add All (version 2025)

Source file name:	Addall.c, Addall.cpp, Addall.java, Addall.py
Input:	Standard
Output:	Standard
Time / Memory limit:	4/9/6 (C++/Java/Python) second(s) / 128 megabytes
Author(s):	Hugo Humberto Morales & Gabriel Gutiérrez - UTP Colombia

Professor Humbertov Moralov, while organizing the papers in his office, found the following solution that he had proposed approximately 10 years ago for the programming challenge **UVa - 10954 - Add All:**

```c
#include <stdio.h>
#define MAXT 5000

int ExtractMin(int A[], int *n)
{
    int temp, i, min;
    for(i = *n - 1; i >= 1; i--)
    {
        if(A[*n] > A[i])
        {
            temp = A[i];
            A[i] = A[*n];
            A[*n] = temp;
        }
    }
    min = A[*n];
    *n = *n - 1;
    return min;
}

void Insert(int A[], int *n, int element)
{
    *n = *n + 1;
    A[*n] = element;
}

int AddAll(int A[], int n)
{
    int result = 0, value1, value2, value3;
    while(n >= 2)
    {
        value1 = ExtractMin(A, &n);
        value2 = ExtractMin(A, &n);
        value3 = value1 + value2;
        result = result + value3;
        Insert(A, &n, value3);
    }
    return result;
}

int main()
{
    int n, A[MAXT + 1], index;
    while(scanf("%d", &n) && (n > 0))
    {
        for(index = 1; index <= n; index++)
            scanf("%d", &A[index]);
        printf("%d\n", AddAll(A, n));
    }
    return 0;
```

```
}
```

In the previous solution, we work with an array $A[\ ]$ in which n elements are initially stored. In the function `AddAll`, within the `while` repetition loop, the two smallest values are extracted (and removed) from the array, added together, and this value is inserted (stored) in the array. This value is also used to update the total sum in the variable `result`, which stores the result of adding all pairs of smallest values in the array as long as it contains 2 or more values. The computational cost of the `AddAll` function is $O(n^2)$, being the number n the amount of elements stored in the array.

The `while` loop is executed $n-1$ times, because for each iteration there is one number less stored on the array. Each iteration has a total cost of $O(n)$, because the array must be traversed to determine the minimum value in other words: $O((n-1)\cdot n) = O(n^2)$.

But $O(n^2)$ is a computationally very expensive solution, and the only reason why the UVa Online Judge currently gives an `accepted` verdict is because it is a very old programming challenge (from 2005), and at that time a size of n=5,000n = 5,000 n=5,000 was sufficient for a time of 1 or 2 seconds with the computing power of the servers of that time.

Nowdays (year 2025) a solution on the order of 10^8 operations runs in 1 or 2 seconds on online judges. For this reason, you are asked as a student of a Computer Science academic program to propose a solution that in the worst case runs in a complexity of $O(n \cdot \log n)$ per test case, where the function `AddAll` needs to be efficiently programmed for the new size of variable n which is now 10^6.

Input

The input begins with a positive integer t $(1 \leq t \leq 5)$, denoting the number of test cases. Each test case consists of two lines. The first line contains a positive integer n $\left(2 \leq n \leq 10^6\right)$, which represents the number of elements to store in the array $A[\ ]$. The second line contains n positive integers (each greater than or equal to 1 and less than or equal to 10^6).

It is guaranteed that the sum of all n values across the t test cases is at most 10^6.

Output

For each test case, print a single line with the result produced by the `AddAll` function.

Example

Input	Output
5 3 1 2 3 4 1 2 3 4 5 5 4 3 2 1 5 3 2 5 1 4 5 1 1 1 1 1	9 19 33 33 12

Use fast I/O methods

Problem B. Blind Queue Dominoes

Source file name:	Blind.c, Blind.cpp, Blind.java, Blind.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/2/2 (C++/Java/Python) second(s) / 64 megabytes`
Author(s):	`Gabriel Gutiérrez Tamayo - UTP Colombia`

Dominoes is a turn-based board game consisting of 28 different tiles, where each tile is identified by two integers between 0 and 6. In this version of the game, two players participate, and their tiles remain face down; they can only see one tile per turn. Before starting a game, each player selects 7 tiles and arranges them in a row from left to right. In each turn, a player flips their leftmost tile to reveal its value and, if either of its two values matches one of the two ends of the play sequence, they can attach it to the corresponding end by the matching side. Otherwise, they place the tile face down again at the beginning, but this time from right to left. After a player places a tile at one of the ends or decides to pass, the other player takes their turn.

A game ends when a player runs out of tiles, or if since the last turn a tile was placed, all remaining tiles have been flipped at least once. When the game ends, the player who has no tiles left wins. If both players still have tiles, the winner is the one with the lowest score.

The **score** of a player at the end of the game is the sum of the points of the unused tiles.

Each player keeps track of how many different tiles they have seen a specific number on, counting double tiles twice. This is called the number counter. Each integer from 0 to 6 has a counter. Both players are very careful not to show their tile to the other player when flipping it, so each player keeps track based only on the tiles they have flipped or that have already been placed.

Alice and Bob want to play some games, and they always follow the same strategy on their turn:

- If both ends are the same and either value of the tile matches the ends, attach the tile to either end.
- If the ends are different and the tile matches only one end, attach it to that end.
- If the ends are different and the tile matches both, place the tile on the end with the lower counter. In case of a tie, place it on the end with the smaller value.

Given the tiles each player has, determine who wins the game. Alice always has the first turn and will always place her first tile on that turn.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$), indicating the number of test cases. Each test case consists of two lines, each containing 7 pairs of integers a_i, b_i ($0 \leq a_i, b_i \leq 6$), describing Alice's tiles in the first line and Bob's tiles in the second line. It is guaranteed that, for each test case, all 14 tiles are distinct.

Output

For each test case, print a line in the format W T A B, where W is a string indicating who won the game, `"Alice"`, `"Bob"` or `"Draw"` in case of a tie, T is the number of turns the game lasted, and A and B are the final scores of Alice and Bob, respectively.

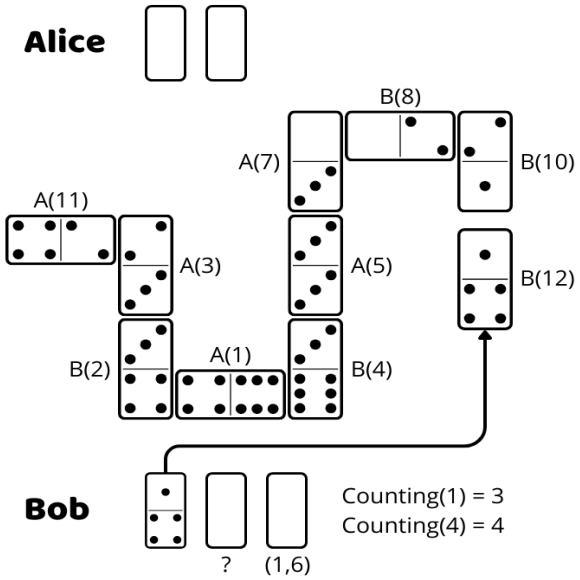


Figure 2. First test case, turn 12

Example

Input	Output
3 4 6 2 3 3 3 0 3 5 3 4 2 6 2 4 3 3 6 1 6 0 2 2 1 1 4 1 1 0 3 1 5 5 4 1 6 4 2 5 0 1 3 3 2 0 1 4 3 4 4 5 3 1 4 2 0 3 5 5 6 3 3 4 1 6 2 2 2 4 4 3 2 6 3 3 4 1 5 6 1 4 2 6 0	Bob 16 16 9 Alice 26 6 16 Draw 14 12 12

Use fast I/O methods

Problem C. Consecutive Reduction

Source file name:	Consecutive.c, Consecutive.cpp, Consecutive.java, Consecutive.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/2/8 (C++/Java/Python) second(s)` / `64 megabytes`
Author(s):	`Gabriel Gutiérrez Tamayo - UTP Colombia`

Consider an infinite array of non-negative integers, where for each position i ($i \geq 0$), $a_i = i$. You are given several queries, each with a range $[l,\ r]$. For each query, you must answer how many operations are needed to make all elements in that range become zero.

An operation consists of selecting an a_i ($l \leq i \leq r,\ a_i \geq 1$). If a_i is even, divide it by 2; if it is odd, subtract 1.

Input

The first line contains a positive integer q $\left(1 \leq q \leq 10^5\right)$, indicating the number of queries. Each query consists of a line with two non-negative integers l r $\left(0 \leq l \leq r \leq 10^{12}\right)$.

Output

For each query, print a line indicating the number of operations needed to convert all the elements in the respective range to zero.

Example

Input	Output
5 0 8 5 7 9 9 2 10 6 13	26 13 5 35 40

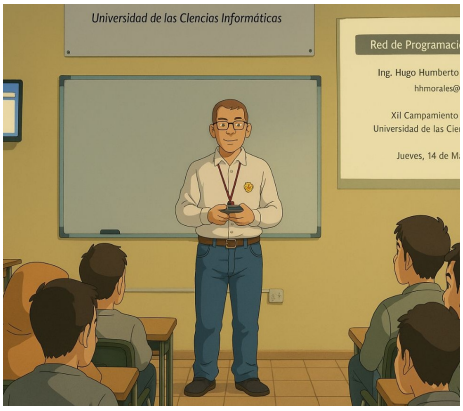
Use fast I/O methods

Problem D. Delete, delete, delete, ...

Source file name:	Delete.c, Delete.cpp, Delete.java, Delete.py
Input:	Standard
Output:	Standard
Time / Memory limit:	1/2/2 (C++/Java/Python) second(s) / 64 megabytes
Author(s):	Hugo Humberto Morales Peña - UTP Colombia

Professor Humbertov Moralov has had plenty of free time during his latest vacation and has spent it writing programming challenges to torture (I mean, to properly evaluate the knowledge) his new programming students.

Professor Moralov has hidden a secret message in a text (that is, in a string of characters). To uncover the hidden message, all occurrences of a given list of characters must be removed from the text. Additionally, it is known that the hidden message does not begin or end with one or more whitespace characters and that it does not contain the same symbol consecutively.



Humbertov Moralov's image by ChatGPT "Studio Ghibli" style

Input

The input contains a single test case consisting of two lines.

The first line contains a text of minimum length 10 and maximum length 10^6, which may include lowercase English letters, digits from 0 to 9, and the space character (ASCII 32).

The second line contains a string of characters (symbols) that do not belong to the message, enclosed between an opening bracket ("[") and a closing bracket ("]"), with a length between 1 and 36. It may contain lowercase English letters, digits from 0 to 9, and even the space character (ASCII 32).

Output

Print in a single line the string representing the hidden message. The line must end with a newline character ("\n").

Note: The shortest hidden message consists of a single letter or digit. Moreover, the hidden message contains no leading or trailing whitespace.

Example

Input
king lokif bfbbddddafbbtggffaf kong bsbtbgrgugfcgtfugrfegg bigbig kjong kiko [bfghijklmno]
Output
data structure

Input
auub attb appb aoob appb aeeb annb a22b a00b a22b a55b [abc]
Output
utpopen2025

Use fast I/O methods

Problem E. Ernesto

Source file name:	Ernesto.c, Ernesto.cpp, Ernesto.java, Ernesto.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`2/3/3 (C++/Java/Python) second(s) / 256 megabytes`
Author(s):	`Santiago Gutiérrez Alzate - Google`

Ernesto is a businessman with a large number of ideas to become a millionaire. The problem is that he doesn't have time to develop them (because he is always busy), so he needs to find students to develop them for him. However, Ernesto has a couple of restrictions:

1. The students must work for him for free, as he is very stingy and doesn't like to pay.
2. He only has time to "exploit" one student at a time.

Balloons in programming contest, *Semillero In Silico, UTP*

Naturally, the offer doesn't seem very appealing to students, so what Ernesto does is hire them without telling them how much they will be paid. This way, they only realize they are working for free when they try to claim their first paycheck.

Since many students are friends with each other, if Ernesto hires a student, this student—after not receiving any payment—will warn all their friends not to work for Ernesto. The friends of this student, in turn, will warn all their other friends, who will warn all their friends that have not yet encountered Ernesto, and so on. Eventually, the entire group of friends and friends-of-friends of the exploited student will be warned. Obviously, there will come a time when all the students will be warned, and Ernesto will not be able to continue hiring.

Since each student has a tolerance to exploitation H (which determines the number of hours they would work without asking for payment), Ernesto has assigned you the task of telling him which students he should hire in order to maximize the number of hours students will work for him. You're in luck: this time the stingy Ernesto has promised you a balloon if you advise him correctly.

Input

The input begins with a positive integer T ($1 \leq T \leq 10$), which represents the total number of test cases. The first line of each test case contains two integers N and M $\left(1 \leq N \leq 10^5,\ 0 \leq M \leq 10^6\right)$, which respectively represent the number of students and the number of friendship relations. The following N lines each contain a string and a positive integer, S H $\left(1 \leq |S| \leq 10,\ 1 \leq H \leq 10^8\right)$, which respectively represent the name of the student and their number of hours of tolerance to exploitation. The student name contains only lowercase letters from the English alphabet. It is guaranteed that there are no two students with the same name, and also no two students with the same number of tolerance hours.

The test case continues with M lines, each containing a pair of strings A B (equivalent to B A, with $A \neq B$, $1 \leq |A|, |B| \leq 10$) indicating that students with names A and B are friends. Each friendship relation will appear at most once per test case.

Note 1: The sum of all N values across all T test cases is at most 10^5.

Note 2: The sum of all M values across all T test cases is at most 10^6.

Output

For each test case, print one line with "`Case idCase:`" — the quotes are for clarity only — where `idCase`

is replaced by the test case number, followed by as many lines as necessary with the names of the students in alphabetical order. These are all the students Ernesto should hire in order to maximize the number of hours students will work for free for him in that test case. For clarity, refer to the sample input and output below.

Example

Input	Output
2 7 4 arturo 100 daniel 15 carlos 50 carolina 20 jorge 1000 jessica 16 susana 1 daniel arturo carlos carolina carolina jessica susana jessica 5 4 sara 10 marcela 20 dario 25 adrian 26 laura 15 marcela sara dario marcela laura dario adrian laura	Case 1: arturo carlos jorge Case 2: adrian

Use fast I/O methods

Problem F. Furthermore ... Add All (version 2025)

Source file name:	Furthermore.c, Furthermore.cpp, Furthermore.java, Furthermore.py
Input:	Standard
Output:	Standard
Time / Memory limit:	3/8/8 (C++/Java/Python) second(s) / 512 megabytes
Author(s):	Hugo Humberto Morales & Gabriel Gutiérrez - UTP Colombia

Professor Humbertov Moralov, while organizing the papers in his office, found the following solution he proposed approximately 10 years ago for the programming challenge **UVa - 10954 - Add All:**

```
#include <stdio.h>
#define MAXT 5000

int ExtractMin(int A[], int *n)
{
    int temp, i, min;
    for(i = *n - 1; i >= 1; i--)
    {
        if(A[*n] > A[i])
        {
            temp = A[i];
            A[i] = A[*n];
            A[*n] = temp;
        }
    }
    min = A[*n];
    *n = *n - 1;
    return min;
}

void Insert(int A[], int *n, int element)
{
    *n = *n + 1;
    A[*n] = element;
}

int AddAll(int A[], int n)
{
    int result = 0, value1, value2, value3;
    while(n >= 2)
    {
        value1 = ExtractMin(A, &n);
        value2 = ExtractMin(A, &n);
        value3 = value1 + value2;
        result = result + value3;
        Insert(A, &n, value3);
    }
    return result;
}

int main()
{
    int n, A[MAXT + 1], index;
    while(scanf("%d", &n) && (n > 0))
    {
        for(index = 1; index <= n; index++)
            scanf("%d", &A[index]);
        printf("%d\n", AddAll(A, n));
    }
    return 0;
```

```
}
```

In the solution above, an array $A[\,]$ is used to initially store n elements. In the function `AddAll`, the `while` loop extracts (and removes) the two smallest values from the array, sums them, and stores that value back into the array. That value is also used to update the total sum in the variable `result`, which stores the result of summing all the pairs of smallest values in the array as long as it contains 2 or more values. The computational cost of the function `AddAll` is $O(n^2)$, where n is the number of elements stored in the array.

The `while` loop runs $n-1$ times because in each iteration the array contains one fewer element. The cost of each iteration is $O(n)$ because all n stored elements in the array $A[\,]$ must be traversed to determine the minimum value. That is, $(n-1) \cdot O(n) = O(n^2)$.

An $O(n^2)$ is a very expensive time complexity, and the only reason why the Online Judge UVa currently gives a verdict of `accepted` is because it is a very old programming challenge (year 2005), and at that time a size of $n = 5{,}000$ was sufficient for a runtime of 1 or 2 seconds with the computing power of that era's servers.

Nowadays (year 2025), a solution of order 10^7 operations runs in under a second on online judges. For that reason, to make the challenge even more interesting, you, as a student in a Computer Science academic program, are asked to propose a solution that, in the worst case, runs in $O(n)$ time per test case. To achieve this, you must efficiently implement the `AddAll` function for the new size limit of the variable n, which is now 10^7.

Input

The input begins with a positive integer t $(1 \leq t \leq 5)$, denoting the number of test cases. Each test case consists of two lines. The first line contains a positive integer n $\left(2 \leq n \leq 10^7\right)$, representing the number of elements to store in the array $A[\,]$. The second line contains n positive integers (each greater than or equal to 1 and less than or equal to 10^5).

It is guaranteed that the sum of all values of n in the t test cases is at most $2 \cdot 10^7$.

Output

For each test case, print a single line with the result generated by the `AddAll` function.

Example

Input	Output
5 3 1 2 3 4 1 2 3 4 5 5 4 3 2 1 5 3 2 5 1 4 5 1 1 1 1 1	9 19 33 33 12

Use fast I/O methods

Problem G. Gerson and the Anagrams

Source file name:	Gerson.c, Gerson.cpp, Gerson.java, Gerson.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/2/2 (C++/Java/Python) second(s) / 64 megabytes`
Author(s):	`Wilmer Emiro Castrillón Calderón - UNIAMAZONIA Colombia`

Two strings are anagrams if both have the same letters, in the same quantity, but in a different order. For example, frase" is an anagram of fresa", but frase" is not an anagram of fruta".

Gerson and Melissa enjoy word games, especially finding anagrams. Gerson has invited you to try his new game. Melissa has written two strings A and B of equal length, and your task is to count, for each substring in A, how many anagrams it has among all the substrings of B, and print the total.

They are not very strict with the definition and consider that a word is also an anagram of itself. That is, for them, two equal substrings are also anagrams.

Input

The input consists of an integer T ($1 \leq T \leq 1000$), the number of test cases. Each test case consists of two lines containing the strings A and B ($1 < |A|,\ |B| \leq 1000,\ |A| = |B|$). It is guaranteed that the strings consist only of lowercase letters of the English alphabet, and the sum of the lengths of all A across all test cases is less than or equal to 1000.

Output

For each test case, print on a single line the total count of anagrams.

Example

Input	Output
3 abc axy abcde baxyz helloworld oxllxxhexx	1 3 12

Use fast I/O methods

Problem H. Humbertov and the Maze

Source file name:	Maze.c, Maze.cpp, Maze.java, Maze.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`2/3/4 (C++/Java/Python) second(s)` / `128 megabytes`
Author(s):	`Hugo Humberto Morales & Gabriel Gutiérrez - UTP Colombia`

One of Professor **Humbertov Moralov**'s passions is the analysis of mazes. He is now interested in identifying the distance of the farthest pair of cells in the largest region of cells within the maze that are walkable between them.

For simplicity, you can consider the maze as a grid (a matrix) of cells. A cell can be a wall or a passage (a cell through which you can walk). At any moment, Professor Moralov can move to a new cell from his current position if they share a side and both are passage cells.

Since you are a student in the Data Structures course, Professor Humbertov Moralov needs your help to solve this challenge. You must write a program to calculate the distance of the farthest pair of walkable cells in the largest region of cells inside the maze through which you can walk between them.

Note: In the case where the maximum region size is shared by several regions, you must calculate which of them has the maximum distance between the farthest pair of cells.

It is guaranteed that for any pair of cells within a passage cell region (a region through which you can walk), there is a unique path between them.

Input

The input begins with a positive integer T ($1 \leq T \leq 10$), denoting the number of test cases.

Each test case starts with a line containing two positive integers H and W (High, Wide, $3 \leq H, W \leq 1000$). The test case continues with H lines, each of which contains W characters. Each character represents the status of a cell in the maze as follows:

1. '`.`' - to indicate that it is a passage cell (a cell through which you can walk).
2. '`#`' - to indicate that it is a wall cell.

Output

For each test case, print a line with the following format: `Case idCase: R T D`, where `idCase` is replaced by the test case number, `R` is replaced by the number of isolated regions of walkable passage cells in the maze, `T` is replaced by the size of the largest independent region of walkable cells, and `D` is the minimum distance between the farthest pair of cells within the largest walkable region. For clarity, refer to the input and output examples below.

Example

Input	Output
3 11 11 #.......#.. ..#.#.#...# #.#.#.##### ..#.#.#.... .#########. ....####.####. ....#.##.#. ###.#.#..#. .#..#.#.##. ...##.#.... 11 11 #...#...#.. ..#.#.#...# #.#.#.##### ..#.#.#.... .#########. .#......... .####.####. ....#.##.#. ###.#.#..#. .#..#.#.##. ...##.#.... 11 11 #...#...#.. ..#.#.#...# #.#.#.##### ..#.#.#.... .########## .#......... .####.####. ....#.####. ###.#.#..#. .#..#.#.##. ...##.#....	Case 1: 1 69 39 Case 2: 3 31 22 Case 3: 4 25 23

Use fast I/O methods

Problem I. Internal Happiness

Source file name:	Internal.c, Internal.cpp, Internal.java, Internal.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/1/1 (C++/Java/Python) second(s) / 64 megabytes`
Author(s):	`Eddy Ramírez Jiménez - UNA & TEC Costa Rica`

Students worldwide are engaging with the STEAM curriculum, which integrates Science, Technology, Engineering, Arts, and Mathematics. In some schools, grades are assigned based on students' performance in these subjects.

At certain institutions, an unusual method is used to announce grades: students are called one by one in alphabetical order to present their grades in front of their classmates. This process reveals each student's grades to the entire group.

This public announcement leads to students experiencing internal emotions of happiness at three distinct levels (though they may not outwardly express it):

- Happy: When exactly 3 out of 5 of their grades are higher than those of all the previous students.
- Happier: When exactly 4 out of 5 of their grades are higher than those of all the previous students.
- Happiest: When all 5 of their grades are higher than those of all the previous students.

Your task is to determine, given the grades of all students in order, how many students felt happy, happier, and happiest when their grades were announced.

Input

A single test case.

The first line contains a integer number n $\left(1 \leq n \leq 10^3\right)$ indicating the number of students at the school.

The next n lines contains five integers each separated by space, denoting the grades of the student. $0 \leq g_i \leq 100$.

Output

Three numbers separated by space on a single line. The first is the number of students that were happy when they received their grade, the second one the number of students who were happier when they received their grade and the third number, the amount of students who were happiest when received their grade.

Example

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

Use fast I/O methods

Problem J. Joining the Team

Source file name:	Joining.c, Joining.cpp, Joining.java, Joining.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/2/8 (C++/Java/Python) seconds / 64 megabytes`
Author(s):	`Eddy Ramírez Jiménez - UNA & TEC Costa Rica`

Coach Drump has a unique personality: he always insists that his team practice by forming special lineups during their training sessions.

Each lineup is considered unique if the sequence of player heights that compose it is different from any other.

In every training session, Coach Drump calls his players one by one to line up. However, he has a particular method for selecting them: he always calls the next tallest or the next shortest player available.

To maximize the variety in his training sessions, the coach aims to form a different lineup every day. He has made it clear that if a lineup is ever repeated, he will resign immediately.

A program is needed that, given the heights of the players, determines how many days will pass before Coach Drump finally releases his team from these quirky training methods. Since this number can be extremely large, the result must be given modulo $10^9 + 7$.

Input

The first line contains a single integer T $(1 \leq T \leq 10)$ indicating the test cases.

The next $2 \cdot T$ lines has the description of each test case.

Each test case consist on two lines, the first with a single number n $\left(1 \leq n \leq 3 \cdot 10^4\right)$ indicating the number of players on Drump's training team.

The next line contains a sequence H with n heights separated by space $\left(1 \leq h_i \leq 3 \cdot 10^4\right)$.

Output

For each test case, the output is a single line with the number of unique lineups modulo $10^9 + 7$

Example

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

Use fast I/O methods

Problem K. K Tasks and the Random Scheduler

Source file name:	KTasks.c, KTasks.cpp, KTasks.java, KTasks.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`1/2/2 (C++/Java/Python) second(s) / 64 megabytes`
Author(s):	`Gabriel Gutiérrez Tamayo - UTP Colombia`

You are a computer science student, and at your university, they still keep an old computer with two processing units. The first uses a standard scheduler from that era for CPU allocation, while the second has a scheduler implemented by former students as part of an Operating Systems course project from several years ago, which has not been modified since then.

The university has made this computer available to students for testing purposes, so every time a student runs their own programs, these are executed exclusively on the second processing unit.

The scheduler developed by the students uses a fairly simple algorithm for CPU allocation and works as follows. It always assigns a fixed CPU time of T_C milliseconds to each process and, when that time is up, it selects another process uniformly at random from all processes currently running, including the one that just released the CPU. A process can also lose its turn before exhausting its assigned CPU time if it makes certain system calls.

As part of a project in your Data Structures course, you have developed a program that must complete K tasks, and you now wish to determine the expected execution time on that computer. However, you have already conducted some tests and know that each task requires T_P $(1 \leq T_P \leq T_C)$ milliseconds of computation to be completed on that computer. Since the behavior of the processes running may vary greatly, you apply a simple and pessimistic approach, assuming there are P other processes running in addition to your program, and that each process uses its full assigned CPU time during its turn. However, your program loses its turn every time it completes a task, since it prints a small result to the screen. You also assume that the time to select the next process and to print to the screen is negligible.

Input

The first line contains an integer q $\left(1 \leq q \leq 10^5\right)$, indicating the number of queries. Each of the following q lines contains 4 integers P, K, T_C, and T_P $(1 \leq P,\ K,\ T_C,\ T_P \leq 10^4,\ T_P \leq T_C)$, indicating the number of previously running processes, the number of tasks your program must complete, the CPU time in milliseconds assigned to each process, and the time in milliseconds your program takes to complete one task, respectively.

Output

For each test case, print a line with a number indicating the expected time in seconds to complete all tasks. For each query, your answer is considered correct if its absolute or relative error does not exceed 10^{-4}.

Example

Input	Output
3 1 1 60 30 2 5 60 30 4 4 50 20	0.09 0.75 0.88

Use fast I/O methods

Problem L. List

Source file name:	List.c, List.cpp, List.java, List.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`5/8/8 (C++/Java/Python) second(s) / 128 megabytes`
Author(s):	`Hugo Humberto Morales Peña - UTP Colombia`

Initially, you have an empty list and are given several queries. These queries are basic operations that can be performed on the *List* data structure, such as `Insert` (insert an element into the list) and `Delete` (remove the first occurrence of an element from the list, if it exists). At any moment, it's also necessary to know the value of the **median** of the integers currently in the list. The task is to process the given queries.

In statistics, the **median** is the number located in the middle position of a list of numbers when they are sorted in ascending order, leaving the same number of values on both sides. In the case of a list with an even length, the median is the average of the two middle numbers.

Input

The input contains a single test case. The first line contains a positive integer Q $\left(1 \leq Q \leq 2 \cdot 10^6\right)$ denoting the number of queries to process, followed by exactly Q lines in the following format:

- 1 V: `Insert` operation with V $\left(0 \leq V \leq 10^6\right)$, inserts the integer V into the list.
- 2 V: `Delete` operation with V $\left(0 \leq V \leq 10^6\right)$, removes the first occurrence of the integer V from the list if it exists. If the list is empty, nothing happens.
- 3: Print on a single line the integer part of the median of the integers currently in the list. If the list is empty, print the message: `Empty!`

Output

For each query of type 3, compute and print on a single line the integer part of the median of the integers currently in the list. If the list is empty, print the message: `Empty!`

Example

Input	Output
12 3 2 5 1 10 1 10 3 2 10 3 1 25 1 20 3 1 25 3	Empty! 10 10 20 22

Use fast I/O methods

Problem M. MST Number

Source file name:	MSTNumber.c, MSTNumber.cpp, MSTNumber.java, MSTNumber.py
Input:	Standard
Output:	Standard
Time / Memory limit:	3/6/6 (C++/Java/Python) second(s) / 128 megabytes
Author(s):	Hugo Humberto Morales & Gabriel Gutiérrez - UTP Colombia

In an introductory Data Structures course in a Computer Science program, students typically begin working with graphs and are introduced to the topic of Minimum Spanning Trees (MST), which can be computed using either Prim's or Kruskal's algorithm.

Let the **MST Number** be the number of minimum spanning trees that can be computed and removed from a graph until it becomes disconnected. In other words, the MST Number is the number of times that someone can repeat the process of generating and then removing a minimum spanning tree from the graph until the graph is no longer connected.

The task is to compute the MST Number and the weight of each minimum spanning tree that can be generated from an undirected graph with weighted edges.

Input

The input contains a single test case consisting of several lines. The first line contains two positive integers N M $\left(2 \leq N \leq 10^4,\ 0 \leq M \leq 10^5\right)$, representing the number of vertices and the number of edges, respectively. Each of the next M lines represents an edge of the graph. Each edge consists of three positive integers V_1 V_2 P $\left(1 \leq V_1, V_2 \leq N,\ V_1 \neq V_2,\ 1 \leq P \leq 10^8\right)$, where V_1 and V_2 are the vertex numbers and P is the weight of the edge.

Note: It is guaranteed that no two different edges have the same weight.

Output

The first line of the output must contain a non-negative integer R representing the **MST Number** of the graph, followed by R lines each representing the weight of the minimum spanning tree that is generated and removed from the graph in the process.

Note: It is guaranteed that for all test cases, the programming challenge has a unique solution.

Example

Input	Output
7 13 1 2 30 1 3 15 1 4 10 2 4 25 2 5 60 2 7 40 3 4 41 3 6 20 4 6 45 4 7 35 5 6 61 5 7 21 6 7 31	2 122 251

Use fast I/O methods

Problem N. Non-Coprime Josephus

Source file name:	Noncoprime.c, Noncoprime.cpp, Noncoprime.java, Noncoprime.py
Input:	`Standard`
Output:	`Standard`
Time / Memory limit:	`4/8/8 (C++/Java/Python) second(s) / 512 megabytes`
Author(s):	`Gabriel Gutiérrez Tamayo - UTP Colombia`

Professor Humbertov Moralov teaches the Data Structures course and, since the exam on linked lists is approaching, he proposes to exempt one of the students through a game in which this topic could be applied. The game proceeds as follows:

- There is a bag with n pieces of paper, each containing a number from 1 to n, where each number appears exactly once.
- Each of the n students draws one piece of paper from the bag, and the number on it identifies the student in the game.
- The student with the number 1 is excluded from the game.
- The remaining students stand in a circle, all facing the center, so that for each student i ($2 \leq i \leq n-1$), the student to their left is $i+1$, and the student n has student 2 to their left.
- Several rounds are played until only one student remains. In each round, a student writes their number on a sheet of paper and passes it to their left. Each time a student receives the sheet, they observe the number on it, compute the GCD (Greatest Common Divisor) between their number and the number on the sheet, and if the result is greater than 1, they leave the game. Otherwise, they pass the sheet to the left. The student who starts in the first round is number 2, and in subsequent rounds, the one to the left of the student who was eliminated starts.

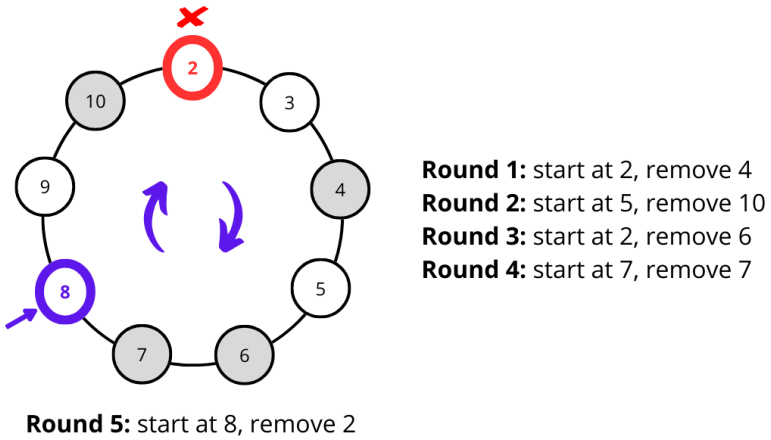


Figure 3. First test case, round 5

Given the number of students, your task is to determine who wins the game.

Input

The first line contains an integer t ($1 \leq t \leq 100$), indicating the number of test cases. Each test case consists of a line with an integer n $\left(3 \leq n \leq 10^6\right)$, indicating the number of students.

It is guaranteed that the sum of all n across test cases is at most 10^6.

Output

For each test case, print a line with a single integer, indicating the number of the student who wins the game.

Example

Input	Output
4 3 7 8 10	3 3 3 8