The 5th UIT-ACM Programming Contest

May 14, 2016

Problems Overview

There are 8 (eight) problems in the packet, using letter A – H.

Problem A – Assembly

Problem B – Caching

Problem C – Factorial

Problem D – Garage

Problem E – Gold Hunt

Problem F – Hereditament

Problem G – Sorting

Problem H – Subsets

Remember: For all problems, read the input data from standard input and write the results to the standard output

Problem A

Assembly

Time limit: 1 second

The Alternate Control Machine (ACM) Factory has a large assembly line to make a type of product. The assembly has N robots ($R_1, R_2, \ldots R_N$) working sequentially. That means a semi-finished product moves from robot R_1, to R_2, then to R_3 ... to R_N. Each robot adds a component to the product. Each robot can complete its own job in P_i products per one hour.

The company has a budget of M VNĐ to improve productivity for the entire assembly. As a product director, you know that robot R_i needs to invest M_i VNĐ to contribute to the production of one more product per hour. You have to optimize the amount of money to invest to each robot to produce maximum number of products per hour.

Input

The first line of input contains one integer T ($1 \leq T \leq 10$) - the number of test cases.

Then T test cases are given as follows:

- The first line of each test case contains an integer N ($1 \leq n \leq 10^5$) and an integer M ($0 \leq M \leq 10^{12}$) – the number of robots and the budget
- Line i-th of the next N lines contains two integers P_i ($1 \leq P \leq 10^9$) and M_i ($1 \leq M_i \leq 10^9$) – information of the robot i-th

Output

For each test case, output in one line the maximum number of products the assembly can make after investing at most M VNĐ.

Sample Input	Sample Output
1 3 7 1 2 2 3 3 1	3

Problem B

Caching

Time limit: 1 second

The infamous problem of caching! John hasn't notice it when his operating system decide which memory page to swap out, or which file to keep in the SSD storage and which go to the larger but slower HDD. But fate has left John in charge of his startup company brand new distributed contain delivery network. In this system data is never where you need it, and fetching data over a network takes time and consumes bandwidth.

The problem can be mitigated by adding a cache, where a node stores some resources locally and if those resources need to be used again, it can simply take them from its cache rather than asking someone else for them. However, caches have a nasty tendency to fill up, so at some point, objects must be evicted from the cache to make room for new objects. Choosing what object to remove from the cache is not easy and there are several different algorithms to choose from.

Lucky for John, his startup is dealing with fortune-tellers and they can see into the future, knowing exactly what objects will be accessed and in what order. Using this information John can make optimal decisions on what objects to remove from the cache. Optimality here means that he will have to minimize the number of times an object is read into the cache.

All object accesses go through the cache, so every time an object is accessed, it must be inserted into the cache if it was not already there. All objects are of equal size, and no writes occur in the system, so a cached object is always valid. When the system starts, the cache is empty.

But can John do that? You have been tasked with evaluating John's algorithm

Input

Input include many test cases, each of which share the following structure:

The first line of input contains three integers, separated by single spaces, telling you how many objects fit in the cache, $0 < c \leq 10^3$, how many different objects are in the system, $c \leq n \leq 10^5$, and how many accesses, $0 \leq a \leq 10^5$, will occur.

The following a lines contain a single integer between 0 and $n - 1$ (inclusive) indicating what object is accessed. The first line corresponds to the first object accessed access and the last line to the last.

Output

Output the least number of times an object must be read into the cache to handle the accesses listed in each test case, follow by new line character.

Sample input	Sample output
1 2 3 0 0 1 3 4 8 0 1 2 3 3 2 1 0	2 5

Problem C

Factorial

Time limit: 1 second

In mathematics, the factorial of a non-negative integer n, denoted by $n!$, is the product of all positive integers less than or equal to n. Some examples of factorial:

n	$n!$
2	2
3	6
4	24
5	120
6	720

As you can see, it gets big quickly. So, we only want to find the first non-zero digit from right to left of $n!$.

Input

The input contains one positive number N $(0 < N < 10000)$

Output

Output a single line the first non-zero digit of $n!$ from right to left.

Sample input	Sample output
3	6
5	2
26	4

Problem D

Garage

Time limit: 1 second

A parking garage has N parking spaces, numbered from 1 to N inclusive. The garage opens empty each morning and operates in the following way throughout the day.

Whenever a car arrives at the garage, the attendants check whether there are any parking spaces available. If there are none, then the car waits at the entrance until a parking space is released. If a parking space is available, or as soon as one becomes available, the car is parked in the available parking space. If there is more than one available parking space, the car will be parked at the space with the smallest number. If more cars arrive while some car is waiting, they all line up in a queue at the entrance, in the order in which they arrived. Then, when a parking space becomes available, the first car in the queue (i.e., the one that arrived the earliest) is parked there.

The cost of parking in dollars is the weight of the car in kilograms multiplied by the specific rate of its parking space. The cost does not depend on how long a car stays in the garage. The garage operator knows that today there will be M cars coming and he knows the order of their arrivals and departures. Help him calculate how many dollars his revenue is going to be today.

Write a program that, given the specific rates of the parking spaces, the weights of the cars and the order in which the cars arrive and depart, determines the total revenue of the garage in dollars.

$1 \leq R_s \leq 100$ The rate of parking space s in dollars per kilogram $1 \leq W_k \leq 10{,}000$ The weight of car k in kilograms

Input

The first line of input contains the integers N ($1 \leq N \leq 100$): The number of parking spaces and M ($1 \leq M \leq 2{,}000$): The number of cars, separated by a space.

The next N lines describe the rates of the parking spaces. The s^{th} of these lines contains a single integer R_s, the rate of parking space number s in dollars per kilogram.

The next M lines describe the weights of the cars. The cars are numbered from 1 to M inclusive in no particular order. The k^{th} of these M lines contains a single integer W_k, the weight of car k in kilograms.

The next $2 * M$ lines describe the arrivals and departures of all cars in chronological order. A positive integer i indicates that car number i arrives at the garage. A negative integer $-i$ indicates that car number i departs from the garage. No car will depart from the garage before it has arrived, and all cars from 1 to M inclusive will appear exactly twice in this sequence, once arriving and once departing. Moreover, no car will depart from the garage before it has parked (i.e., no car will leave while waiting on the queue).

Output

Output a single line containing a single integer: the total number of dollars that will be earned by the garage operator today.

Sample input	Sample output
3 4 2 3 5 200 100 300 800 3 2 -3 1 4 -4 -2 -1	5300

Explanation for sample

Car number 3 goes to space number 1 and pays 300 * 2 = 600 dollars.

Car number 2 goes to space number 2 and pays 100 * 3 = 300 dollars.

Car number 1 goes to space number 1 (which was released by car number 3) and pays 200 * 2 = 400 dollars.

Car number 4 goes to space number 3 (the last remaining) and pays 800 * 5 = 4,000 dollars.

Problem E

Gold Hunt

Time limit: 1 second

Jane is a thirty-something business woman who just recently discover programming. He designs to re-create his favorite childhood game. It's a very simple text based adventure game where you walk around and try to find treasure, avoiding falling into traps. The game is played on a rectangular grid and the player gets very limited information about her surroundings.

The game will consist of the player moving around on the grid for as long as she likes (or until she falls into a trap). The player can move up, down, left and right (but not diagonally). She will pick up gold if she walks into the same square as the gold is. If the player stands next to (i.e., immediately up, down, left, or right of) one or more traps, she will "sense a draft" but will not know from what direction the draft comes, or how many traps she's near. If she tries to walk into a square containing a wall, she will notice that there is a wall in that direction and remain in the position where she was.

For scoring purposes, we want to show the player how much gold she could have gotten safely. That is, how much gold can a player get playing with an optimal strategy and always being sure that the square she walked into was safe, no risk taken. The player does not have access to the map and the maps are randomly generated for each game so she has no previous knowledge of the game.

Input

The first line of input contains two positive integers W and H, neither of them smaller than 3 or larger than 50, giving the width and the height of the map, respectively. The next H lines contain W characters each, giving the map. The symbols that may occur in a map are as follows:

P – the player's starting position

G – a piece of gold

T – a trap

– a wall

. – normal floor

There will be exactly one 'P' in the map, and the border of the map will always contain walls.

Output

Output the number of pieces of gold the player can get without risking falling into a trap.

Sample input	Sample output
7 4 ####### #P.GTG# #..TGG# #######	1
8 6 ######## #...GTG# #..PG.G# #...G#G# #..TG.G# ########	4

Problem F

Hereditament

Time limit: 1 second

A farmer has a land in shape of rectangle has size $n \times m$. He wants to divides his land to give to his k sons (labeled from 1 to k). Dividing process splits the land into smaller equal squares with length 1.

At first, each son will choose 1 square (No one choose same square as others). One son will expand his land by adding others square, which have same edge with their squares collected before in turn by their label until all squares are owned. It means that, they will turn around by their label to expand their land: Son with label 1 will choose first, continue with label 2, ... After label k chose, it turns again to label 1 until all squares are chosen.

Taking it faster, you suggest that you will determine number squares of each son has after this process.

Input

The first line of the input contains three positive integers n and m and k, where $n, m \leq 10^5$ is the size of the land, and $2 \leq k \leq 10$ is number of son. Next k lines each line i^{th} contain 2 positive integers u and v that describe the location of first square of son i^{th} by index of row and column.

Output

Output k respective lines with number of squares of each son.

Sample input	Sample output
4 7 3 1 2 3 6 4 3	11 12 5

Explanation for sample

1	1*	1	1	1	2	2
1	1	1	1	2	2	2
1	1	3	2	2	2*	2
3	3	3*	3	2	2	2

Problem G

Sorting

Time limit: 1 second

Sorting is a common operation in many applications, and efficient algorithms to perform it have been developed. The most common uses of sorted sequences are: making lookup or search efficient; making merging of sequences efficient.

In this task the possible key values are the integers 1, 2 and 3. The required sorting order is non-decreasing. Sorting has to be accomplished by a sequence of exchange operations. An exchange operation, defined by two position numbers p and q, exchanges the elements in positions p and q.

You are given a sequence of key values. Write a program that computes the minimal number of exchange operations that are necessary to make the sequence sorted.

Input

The first line of input contains the number of records N ($1 \leq N \leq 10^5$). Each of the following N lines contains a key value.

Output

Output the minimal number L of exchange operations needed to make the sequence sorted.

Sample input	Sample output
9 2 2 1 3 3 3 2 3 1	4

Problem H

Subsets

Time limit: 1 second

Given 2 integers set $X = \{x_1, x_2, \dots, x_m\}$ and $Y = \{y_1, y_2, \dots, y_n\}$, with:

$$x_1 < x_2 < x_m$$

$$y_1 < y_2 < y_n$$

We called that X has smaller order than Y if:

- Exist i that $x_1 = y_1, x_2 = y_2, \dots, x_{i-1} = y_{i-1}, x_i < y_i$

Or

- $m < n$ and $x_1 = y_1, x_2 = y_2, \dots, x_m = y_m$

Assume that we have a set of positive integer $\{1, 2, \dots, n\}$. From this set, we can create subsets of it. With each subset, we sort all elements by ascending order, rearrange all subsets from smaller to larger and index from 1 to $2^{n-1} - 1$.

Example with $n = 3$:

Index	Subset
1	{1}
2	{1, 2}
3	{1, 2, 3}
4	{1, 3}
5	{2}
6	{2, 3}
7	{3}

Input

The input consists of two positive integer $1 < n \leq 60$ and $1 \leq k \leq 2^{60} - 1$.

Your task is determine subset have index k of set has n elements $\{1, 2, 3, \dots, n\}$.

Output

Output sequence of positive integers separate by one space represent the subset.

Sample input	Sample output
3 2	1 2
4 9	2

This is end of the problems packet

Good Luck !