Colombian Collegiate Programming League

CCPL 2018

Round 9 – August 11

Problems

This set contains 10 problems; pages 1 to 13.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - Timsko

Source file name: `timsko.c`, `timsko.cpp`, `timsko.java`, *or* `timsko.py`

Every year, the University of Zagreb organizes a student team competition in informatics. Each team consists of three students.

Traditionally, the best competitors from the university are girls, and they outnumber boys significantly. This year, boys have raised their voice and a rule was made that each team must consist of exactly one boy and two girls.

To make competitors' lives a little more difficult, the dean of the university has decided to send K of the competitors on an internship in a distant country. Those competitors will not be able to compete.

Given the number of female competitors M, the number of male competitors N, and the number of competitors which have to be sent on an internship K, the dean has to create the maximum number of teams which will be able to attend the competition.

For example, if M is 6, N is 3 and K is 2, the dean can send one girl and one boy on an internship, which leaves him with 5 girls and 2 boys. He can then create two teams from them (one girl is left without a team).

Input

The input consists of several test cases. The first line of each test case contains three integers separated by single spaces: M ($0 \leq M \leq 100$), the number of girls, N ($0 \leq N \leq 100$), the number of boys, and K ($0 \leq K \leq M + N$), the number of competitors which have to be sent on an internship.

The input must be read from standard input.

Output

For each test case, a line must be printed. The line must contain only one number: the maximum number of teams which can be formed.

The output must be written to standard output.

Sample Input	Sample Output
6 3 2 2 1 1 6 10 3	2 0 3

B - Profesor

Source file name: `profesor.c`, `profesor.cpp`, `profesor.java`, *or* `profesor.py`

In a long classroom, N desks are arranged in a single row, with two students sitting at each desk. Students are cranky because they are about to have an art class, and their professor is planning to examine them.

Each student has studied art, but only to a certain level. The old professor can tell by the looks on their faces just how much they have studied. The professor, being an artist, uses a different coloured pencil for each grade. Unfortunately, today he brought only one pencil.

In order to make the examination seem fair, he wants to choose two desks and question one student from each desk positioned between the two desks he has chosen (including the chosen desks). It is important that all examined students deserve the same grades, so he can write them down using his only pencil.

The professor wants to know the maximum number of students he can examine this way, as well as which grade the students will get.

Input

The input contains several test cases. For each test case the first line contains a single integer N ($1 \leq N \leq 100\,000$). Each of the following N rows contains two integers: A_i and B_i, grades deserved by students sitting at desk i ($1 \leq A_i, B_i \leq 5$).

The input must be read from standard input.

Output

For each test case in the input, the a line must be printed. The line must contain two numbers separated by a single space: the maximum number of students the professor can examine and the grade those students will get.

If there are multiple solutions possible, output the one with the smallest grade.

The output must be written to standard output.

Sample Input	Sample Output
1 1 5 3 3 5 4 5 1 3 4 2 1 3 2 5 3 2 5	1 1 2 5 2 2

C - Sretan

Source file name: `sretam.c`, `sretam.cpp`, `sretam.java`, *or* `sretam.py`

Digits 4 and 7 are lucky, while all others are unlucky. An integer is lucky if it contains only lucky digits in decimal notation. We would like to know the K-th lucky positive integer.

Input

Input will consist of a series of lines, each line contains a positive integer K ($1 \leq K \leq 10^9$).

The input must be read from standard input.

Output

Output will consist of a series of lines, one for each line of the input. Each line of output must contain the K-th lucky positive integer.

The output must be written to standard output.

Sample Input	Sample Output
1 2 3	4 7 44

D - Ljutnja

Source file name: `ljutnja.c`, `ljutnja.cpp`, `ljutnja.java`, *or* `ljutnja.py`

Children in a kindergarten have received a large sack containing M candies. It has been decided that the candies are to be distributed among N children.

Each child has stated the number of candies that it wants. If a child isn't given the amount of candy it wants, it will get angry. In fact it'll get angrier for each candy it is deprived of. Some speculate that it's anger will be equal to the square of the number of candy it is deprived of. For instance, if Mirko states that he wants 32 candies but receives only 29, he would be missing 3 candies, so his anger would be equal to 9.

Unfortunately, there is an insufficient amount of candy to satisfy all children. Therefore, the candies should be distributed in such a way that the sum of the children's anger is minimal.

Input

The input consists of several test cases. Each test case consists of several lines: the first line contains two integers, M ($1 \leq M \leq 2 \times 10^9$) and N ($1 \leq N \leq 100\,000$).

The following N lines contain integers (one per line) which represent the wishes of the children. Those numbers are all strictly less than 2×10^9), and their sum always exceeds M.

The input must be read from standard input.

Output

For each case, your program should output a line with the minimum sum of the children's anger.

The output must be written to standard output.

Sample Input	Sample Output
5 3 1 3 2 10 4 4 5 2 3	1 4

E - Tabovi

Source file name: `tabovi.c`, `tabovi.cpp`, `tabovi.java`, *or* `tabovi.py`

Zvonkec is yet another programmer employed in a small company. Every day he has to refactor one file of source code. Much to his dismay, the source is usually far from being clear and tidy. He is especially bothered by uneven indentation, i.e. the number of tabulators (tabs) indenting each line. Fortunately, his editor has a command to select a group of consecutive lines and add or delete a character from the start of each one. Help Zvonkec tidy up the code as quickly as possible.

You are given the number of lines N, a sequence specifying the current number of tabs at the start of each line, and a sequence specifying the required number of tabs at the start of each line.

Zvonkec can execute any number of commands consisting of:

- selecting any number of consecutive lines.
- adding or deleting a single tab to/from the front of each of the selected lines.

The two actions above comprise a single command, regardless of the number of selected lines.

It should be noted that it is forbidden to delete more tabs from a line than are actually present at the start of a line, as the editor would start deleting characters other than tabs.

You are asked to calculate the minimum number of commands required to tidy up the code.

Input

The input consists of several test cases. Each test case consists of three lines:

- The first line contains a positive integer N ($N \leq 1\,000$).
- The second line contains a sequence of N integers P_i ($0 \leq P_i \leq 80$), specifying the number of tabs at the start of i-th line before any editing.
- The third line contains a sequence of N integers K_i ($0 \leq K_i \leq 80$), specifying the number of tabs that Zvonkec would like at the start of i-th line.

The input must be read from standard input.

Output

For each test case, the first and only line in the output must contain the required number, as specified in the problem statement.

The output must be written to standard output.

Sample Input	Sample Output
3 3 4 5 6 7 8 4 1 2 3 4 3 1 1 0 4 5 4 5 5 1 5 0 1	3 6 10

F - Puz

Source file name: `puz.c`, `puz.cpp`, `puz.java`, *or* `puz.py`

There is a snail on the ground. It wants to climb to the top of a wooden pole with the height of V meters, measuring from the ground level. In one day it can climb A meters upwards, however during each night it sleeps, sliding B meters back down. Determine the number of days it needs to climb to the top.

Input

The input consists of several test cases. The first and only line of the case contains three integers separated by a single space: A, B, and V ($1 \leq B < A \leq V \leq 10^9$), with meanings described above.

The input must be read from standard input.

Output

For each test case, print the number of days that the snail needs to reach the top.

The output must be written to standard output.

Sample Input	Sample Output
2 1 5 5 1 6 100 99 1000000000	4 2 999999901

G - Napor

Source file name: `napor.c`, `napor.cpp`, `napor.java`, *or* `napor.py`

Little Mirko wasn't paying attention in math class, so the teacher has decided to give him a tedious assignment to solve during the weekend.

The teacher has given him a text consisting of N lines, containing only digits and lower case letters of the English alphabet. Mirko has to find all numbers in the text and print them out in a nondecreasing sequence. He also has to omit any leading zeros that the numbers may have in the text.

The numbers can be uniquely determined by scanning through the text and always taking the largest possible number, i.e., delimited only by letters or line beginnings/ends. For example, the solution of 01a2b3456cde478 is 1, 2, 478, 3456.

Since Mirko is as slow as the snail from the previous task, he has asked you to write him a program to quickly solve his assignment, so that he can go play with Slavko as soon as possible.

Input

The input contains several scenarios. For each case the first line of input contains the integer N ($1 \leq N \leq 100$), the number of lines of the text.

The next N lines contain the text, consisting exclusively of lowercase English letters and decimal digits.

Each line of the text is at most 100 characters long.

The input must be read from standard input.

Output

For each case the output must contain M lines, where M is the number of numbers found in the provided text. Each line must contain a single number from the text. The numbers must be arranged in a nondecreasing sequence.

The output must be written to standard output.

Sample Input	Sample Output
2 lo3za4 01 4 43silos0 zita002 le2sim 231233 4 01bond 02james007 03bond 04austinpowers000	1 3 4 0 2 2 43 231233 0 1 2 3 4 7

H - Igra

Source file name: `igra.c`, `igra.cpp`, `igra.java`, *or* `igra.py`

Having solved the tedious assignment, Mirko decided to play a game with his good friend Slavko.

They have written a sequence of N letters on a piece of paper. Each one of them is trying to put together a word using letters from the sequence. They alternate taking turns consisting of removing a single letter from the sequence and appending it to the end of their word. Mirko has the first turn. The game ends when no letters are remaining in the sequence.

We define a word to be more beautiful than another word if it comes first alphabetically. The player who has the more beautiful word at the end of the game wins. If both players have equal words, they both lose.

Mirko is a much better player than Slavko, so he has decided to make it easier for Slavko by always selecting the rightmost remaining letter in the sequence. Knowing this, Slavko wants to find out if it is possible for him to win and which is the most beautiful word he can end the game with.

Input

The input contains several cases. For each case the first line of input contains an even positive integer N ($2 \leq N \leq 100\,000$); and the second line contains N characters, the starting letter sequence. All characters are lower case letters from the English alphabet.

The input must be read from standard input.

Output

For each case, the first line of output must contain `DA` if it is possible for Slavko to win, and `NE` otherwise. The second line of output must contain the most beautiful word that Slavko can have at the end of the game.

The output must be written to standard output.

Sample Input	Sample Output
2 ne 4 kava 8 cokolada	NE n DA ak DA acko

I - Knjige

Source file name: `knjige.c`, `knjige.cpp`, `knjige.java`, *or* `knjige.py`

Mirko has a home library consisting of N books arranged one on top of the other in a narrow cabinet. Since being well trained in the secrets of alphabet in the previous task, he now wishes to arrange the books alphabetically, so that the book whose title comes first alphabetically ends up on top, and the alphabetically last one at the bottom of the pile.

Mirko can easily pull a book out of the cabinet, but it is difficult to push it back into the pile, so the book can only be returned to the top of the pile. Thus, the only available method of sorting the books is repeatedly pulling a book out of the pile and placing it on top of the pile.

The books are labelled with integers from 1 to N, in alphabetical order. Therefore, Mirko wants them to be ordered as $(1, 2, \ldots, N)$, counting from the top. For example, if $N = 3$ and the starting order is $(3, 2, 1)$, two moves are sufficient. First, he pulls out the book number 2 and places it on top, so the pile becomes $(2, 3, 1)$. After that, he does the same with book number 1, thus the pile becomes $(1, 2, 3)$.

Help Mirko by calculating the minimum number of moves needed to sort a given starting order.

Input

For each case the first line of input contains the integer N ($N \leq 300\,000$). Each of the next N lines contains a single positive integer. These N integers represent the order of Mirko's books from top to bottom of the cabinet. Each of the integers $1, 2, \ldots, N$ appears exactly once.

The input must be read from standard input.

Output

For each case, output must contain the required minimum number of moves.

The output must be written to standard output.

Sample Input	Sample Output
3 3 2 1 4 1 3 4 2	2 2

J - Crni

Source file name: `crni.c`, `crni.cpp`, `crni.java`, *or* `crni.py`

Even though he has found all the most amusing rides, Mirko's enthusiasm still isn't fading. He opened his graph paper notebook and started colouring squares, and a new, even harder problem dawned on him.

You are given a square table consisting of N rows by N columns. Each cell is either black or white.

A set of cells forming a rectangle, with horizontal and vertical edges following cell borders, shall be called a black rectangle if all cells inside the rectangle are black and it consists of at least two cells.

Calculate the number of possible selections of two black rectangles that have no common cells. As the required number can be extremely large, you should output the remainder of dividing that number by 10007.

Input

The input contains several sets of inputs. For each case the first line of input contains the integer N ($2 \leq N \leq 1\,000$). Each of the next N lines contains a single row of the table, consisting of N symbols. The symbol `C` represents a black cell, while `B` represents a white cell.

The input must be read from standard input.

Output

For each case the first and only line of output must contain the remainder of dividing the required number by 10007.

The output must be written to standard output.

Sample Input	Sample Output
2 CC CC 3 CCB CCB CBB 5 BCCBB BBCBB BCCBB BBBBB CCBBB	2 5 8