ICPC Latin American Regional Contests – 2024

November 9, 2024

Time Limits for the Warmup Session[1]

Problem	Name	Time Limit Per Test*
A	Blackboard Game	0.5
B	Daily Trips	0.5
C	Deciphering WordWhiz	0.5

```
Output size limit for all problems: 1MB

      If your submission generates more output than this
      within the problem's time limit it will receive a Runtime Error verdict.
      Note that output to stderr also counts towards this limit.


* Note that this year time limits do not depend on the programming language.
```

[1]Times are given in seconds.

icpc.foundation

ICPC International Collegiate Programming Contest // 2024-2025

The 2024 ICPC Latin America Contests

JETBRAINS
icpc global sponsor programming tools

HUAWEI
icpc diamond multi-regional sponsor

ICPC Latin American Regional Contests – 2024

November 09, 2024

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 7.

This problem set is used in simultaneous contests with the following participating countries:

Antigua y Barbuda, Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador, Guatemala, México, Perú, Puerto Rico, República Dominicana, Trinidad y Tobago, Uruguay and Venezuela

v1.0

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.

Problem A – Blackboard Game

Carlinhos and Equalizer are playing a game. The game begins with $3N$ elements, which are integer numbers, written on a blackboard. Then, for N rounds, the following two steps are repeated.

1. Carlinhos, the first player, selects an unchosen element and marks it with a red circle.
2. Equalizer, the second player, picks two unchosen elements, marks one of them with a blue square, and erases the other from the blackboard.

At the end of these rounds, the blackboard contains N red-marked elements and N blue-marked elements, with no moves left. The game concludes with a clear winner: if the sum of the red-marked elements differs from the sum of the blue-marked elements, Carlinhos emerges victorious; otherwise, Equalizer takes the win.

The figure below depicts the only possible outcome for the first sample. In this case Equalizer wins for sure, no matter how they play both sums will be equal to 25.

Carlinhos, feeling the game is imbalanced, seeks to determine whether he can secure a victory when both players play optimally. Can you help him with this task?

Input

The first line contains an integer N ($1 \leq N \leq 1000$).

The second line contains $3N$ integers $B_1, B_2, \ldots, B_{3N}$ ($-10^5 \leq B_i \leq 10^5$ for $i = 1, 2, \ldots, 3N$), representing the numbers initially written on the blackboard.

Output

Output a single line with the uppercase letter "Y" if Carlinhos can win the game and the uppercase letter "N" otherwise, assuming both players play optimally.

Sample input 1	**Sample output 1**
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	N

Sample input 2	**Sample output 2**
2 1 2 4 8 16 32	Y

Explanation of sample 2:
Carlinhos wins no matter how he plays, since all subsets have distinct sums.

Sample input 3	Sample output 3

`1`
`2 3 3`
	`Y`

Explanation of sample 3:
Carlinhos can win by picking the number 2. Notice that he would have lost if he picked a 3.

Problem B – Daily Trips

Bella is a simple girl with a simple life: wake up, go to work, work, go home, rest, sleep, and repeat. Bella commutes via bus, and it rains often in her city, so sometimes she needs an umbrella. However, the local weather forecast is unreliable, so Bella can't be sure if it's going to rain or not until right before she begins a trip. To avoid getting caught unprepared, Bella created a system.

She owns one or more umbrellas, keeping them at home or her workplace. Before any trip (from home to work, or vice versa), Bella looks outside to decide whether to bring an umbrella on that trip:

- if it's raining, she brings an umbrella;
- otherwise, if there are currently no umbrellas at her destination (either work or home), she still brings an umbrella, just in case;
- otherwise, she doesn't bring an umbrella.

The second rule above is meant to prevent a situation where Bella needs an umbrella but has none at her current location (a bad memory she will talk about to anyone who asks). This guarantees that Bella will never catch rain and get sick.

Now we need you to simulate Bella's method for a certain period. The simulation starts with Bella at home. Each day she takes two bus trips: to and back from work. Given the starting numbers of umbrellas at her home and workplace, and the weather reports during N consecutive days, find out whether or not Bella brought an umbrella on each of her $2N$ bus trips.

Input

The first line contains three integers N $(1 \leq N \leq 10^4)$, H $(1 \leq H \leq 100)$, and W $(0 \leq W \leq 100)$, indicating respectively the duration of the simulation period in days, and the starting numbers of umbrellas at Bella's home and workplace. For $i = 1, 2, \ldots, N$, the i-th of the next N lines contains two characters representing whether it rained on each trip of the i-th day. The first character refers to the first trip of the day (from home to work), while the second character refers to the second trip of the day (from work to home). Each character is the uppercase letter "Y" if it rained, and the uppercase letter "N" otherwise.

Output

Output N lines. For $i = 1, 2, \ldots, N$, the i-th line must contain two characters indicating whether Bella brought an umbrella on each trip of the i-th day. The first character refers to the first trip while the second character refers to the second trip. Each character must be the uppercase letter "Y" if Bella brought an umbrella, and the uppercase letter "N" otherwise.

Sample input 1	Sample output 1
5 2 1 Y N N N Y N N Y Y Y	Y N N N Y Y N Y Y Y

This page would be intentionally left blank if we would not wish to inform about that.

Problem C – Deciphering WordWhiz

WordWhiz is a popular word puzzle game that challenges players to guess a secret word within a limited number of attempts. The game uses a dictionary containing N words. Each word in this dictionary consists of five distinct lowercase letters.

The game begins with the player being presented with an empty grid, consisting of a number of rows. Each row allows a single guess. The player's task is to fill rows with words contained in the dictionary until the secret word is found, or the player has used all available rows.

After the player submits a guess, the game provides feedback by coloring the cells where the guess was written. The feedback consists of three colors:

- Gray ("X"): The letter in the cell is not part of the secret word.
- Yellow ("!"): The letter in the cell is part of the secret word but is in the wrong position.
- Green ("*"): The letter in the cell is part of the secret word and is in the correct position.

To illustrate, let's consider the scenario where the secret word is "`hotel`", and the player submits "`blast`" as their guess. In this case, the first, third, and fourth cells would turn gray because "`b`", "`a`", and "`s`" are not present in the secret word "`hotel`". The second and fifth cells, however, would turn yellow. This indicates that "`l`" and "`t`" are part of the secret word but appear in wrong positions: "`l`" should be in the fifth position instead of the second, while "`t`" should be in the third position instead of the fifth. This feedback would be represented by "`X!XX!`".

Now, if the player submits "`heart`" as their guess, the third and fourth cells would still turn gray, because "`a`" and "`r`" are not in "`hotel`". The second and fifth cells would again turn yellow, because once more "`t`" is in the fifth position (instead of the third), and this time "`e`" is in the second position when it should be in the fourth. However, for this guess the first cell would turn green, indicating that "`h`" is the first letter in both the guess "`heart`" and the secret word "`hotel`". This feedback would be represented by "`*!XX!`".

Finally, if the player submits "`hotel`" as their guess, all cells would turn green since this is the secret word. This feedback would be represented by "`*****`".

The feedbacks above can be seen in the following picture.

b l a s t
h e a r t
h o t e l

Some time ago, your company added a WordWhiz player on its website and now wants to enhance the game by adding functionality to display previous game sessions. However, only the feedback for each guess was stored, not the submitted words. This means that it might not be possible to accurately recover the guesses submitted in each session, and before investing any further effort, the company wants to analyze the recorded game sessions.

Given a dictionary of five-letter words, the secret word (included in the dictionary) and the feedback for a game session, your task is to determine how many words in the dictionary could have been submitted as each guess.

Input

The first line contains an integer N ($1 \leq N \leq 1000$) indicating the number of words in the dictionary.

Each of the next N lines contains a string representing a word in the dictionary. All strings are different and each of them consists of five different lowercase letters. The first string is the secret word for the game session.

The next line contains an integer G ($1 \leq G \leq 10$) indicating the number of guesses during the game session.

Each of the next G lines contains a five-character string representing the feedback for a guess. The feedback string contains only the characters "`X`", "`!`" and "`*`", indicating respectively gray, yellow and green colors.

It is guaranteed that the input describes a valid game session.

Output

Output G lines, such that the i-th contains an integer indicating how many words in the dictionary could have been submitted on the i-th guess.

Sample input 1	**Sample output 1**

```
6
hotel
weary
heart
blast
pilot
vague
3
X!XX!
*!XX!
*****
```
	```
	1
	1
	1
	```

Explanation of sample 1:
The only possibility is that the player submitted guesses as described in the statement.

Sample input 2	Sample output 2

```
3
scale
table
maple
5
X!X**
X!X**
X!X**
X!X**
X!X**
```
	```
	2
	2
	2
	2
	2
	```

Explanation of sample 2:
The feedback when either "`table`" or "`maple`" is submitted as a guess is "`X!X**`" (because the secret word is "`scale`"). This means that for this game session, the player could have submitted either of these words for each attempt.

Sample input 3	Sample output 3

```
4
scale
table
maple
smile
4
X!X**
*XX**
X!X**
*****
```
	```
	2
	1
	2
	1
	```

Sample input 4	Sample output 4

```
5
latin
mrica
think
solve
debug
1
*****
```
	`1`

ICPC Latin American Regional Contests – 2024

Contest Session: November 9, 2024

Testing Environment and Submission System

1 Information on the Testing Environment

1.1 Environment

The submission correction system will run on the `Ubuntu GNU/Linux 22.04 LTS amd64` distribution, with the following compilers and interpreters configured:

```
C: gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)
C++20: gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)
Python: Python 3.10.12
Java: openjdk 17.0.11
Kotlin: kotlin 1.9.10 (JRE 17.0.11+9-Ubuntu-122.04.1)
```

1.2 Memory limits

```
C, C++20, Python: 1GB
Java, Kotlin: 1GB + 100MB stack
```

1.3 Time limits

Before the contest, the judges will have solved all problems in languages from at least two of the three distinct language groups (C/C++, Java/Kotlin, and Python3). Time limits for each problem will be calculated based on the runtime of those solutions.

A link to the document containing time limits for each problem will be available on the `Problems` tab of Boca's web interface.

1.4 Other limits

```
Source file size: 100KB
Output size limit: Specified per problem, alongside the time limits.
```

1.5 Compilation commands

```
C: gcc -x c -g -O2 -std=gnu11 -static -lm
C++20: g++ -x c++ -g -O2 -std=gnu++20 -static
Java: javac
Python: python3 -m py_compile
Kotlin: kotlinc -J-Xms1024m -J-Xmx1024m -J-Xss100m -include-runtime
```

1.6 C/C++20

- Your program must return a zero, executing, as the last command, `return 0` or `exit(0)`.
- It is known that for problems with very large inputs, `iostream` objects can be slow as, by default, they use a buffer synchronized with the stdio library. If you want to use `cin` and `cout`, disabling such synchronization is advised. This can be achieved by calling `std::ios::sync_with_stdio(false);` at the beginning of your main function. Note that, in this case, using `scanf` and `printf` in the same program should be avoided, since, with separate buffers, misbehavior might occur.

1.7 Java

- DO NOT declare a `package` in your Java program; if you declare a package the program will not execute in Boca.
- Notice that the convention for the solution file name must be obeyed, which means that your public class name must be a capital letter (A, B, C, ...).
- Command for running a Java solution: `java -Xms1024m -Xmx1024m -Xss100m {problem_code}`

1.8 Python

- Note that only Python 3 is supported.
- Python programs will be "syntax checked" when submitted; programs which fail the syntax check will receive a "Compilation Error" verdict.
- Please be aware that Python solutions may not be able to meet the time constraints for some problems. While Python is a versatile and user-friendly language, it may not always provide the best runtime performance for certain tasks.

1.9 Kotlin

- DO NOT declare a `package` in your Kotlin program; if you declare a package the program will not execute in Boca.
- Notice that the convention for the solution file name must be obeyed, which means that your source file must be named (A.kt, B.kt, C.kt, ...).
- Command for running a Kotlin solution: `java -Xms1024m -Xmx1024m -Xss100m {problem_code}Kt`

2 Instructions for the Usage of the Boca Submission System

2.1 Submission of Solutions

To submit a solution, you must use the Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Runs`. Choose the appropriate problem, the language used and upload the source file.

The verdicts you may receive from the judges are:

- 1 - YES
- 2 - NO - Compilation error
- 3 - NO - Runtime error
- 4 - NO - Time limit exceeded
- 5 - NO - Wrong answer
- 6 - NO - Contact staff

Meanings of 1, 2, 3, 4 and 5 are obvious. 6 is used for unforeseeable circumstances. In this case, use the "Clarifications" menu and provide the "run" number for further clarification.

Your program may be run on multiple input files. Note that this means that if your program has more than one error (say, Time Limit Exceeded and Wrong Answer), then you can get either error as verdict.

There is no such thing as "Presentation Error" or "Format Error". If you misspell the word "impossible", for example, and the problem requires that word as output, then your submission will be judged as "Wrong Answer".

Output formatting should follow the sample output in the problem statement, although extra whitespace within reason is acceptable. For example, if you print out thousands of blank spaces within the time and output limit of the problem, that would be judged as a Wrong Answer; however an extra blank at the end of a line or an extra blank line is acceptable.

If your problem outputs more content than the specified **`Output size limit`**, your submission will receive a Runtime Error verdict. Note that any content written to the standard error stream also counts toward this limit.

Penalties

Each submission to a problem that receives a "`NO`" verdict before the first "**`YES`**" verdict for the same problem will incur a time penalty of 20 minutes that is added to the total time of the team. There is no penalty time for unsolved problems.

Exceptions for this rule are "**`NO - Compilation error`**" and "**`NO - Contact staff`**" that have no time penalty associated with them.

Response Times

Note that the time required for coming up with a verdict varies depending on the problem, the verdict, and the point at which the contest is in when the submission is received. As solutions are judged simultaneously, this means that you might receive the verdict of your runs out of order. Also, in some cases, manual verification is required from the judges. Depending on what needs to be verified and the number of submissions that require manual verification, even delays on the order of minutes are expected.

2.2 Clarifications

All communication with the judges is done through clarification messages.

To request a clarification concerning a problem statement, you must use Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Clarifications`. Choose the appropriate problem and type your question.

Both clarification replies from the judges and requests sent by you are displayed there. There will be an alert once your clarification is replied (or the judges issue a global clarification).

Note that it's quite uncommon to have clarifications issued and most of the time the answer to the questions received is already on the problem statement. Please read the problem statement and examine the sample test cases carefully before submitting any request for clarifications.

2.3 Score Board

To visualize the score board showing the teams ranking, you must use Boca's web interface:

- Open your browser.
- Login as a team (username and password assigned to your team).
- Access the tab `Score`. You will have access to the local score board.

2.4 Tasks

The tab `Tasks` allows the team to send files for printing, as well as ask for help from a staff member.

- To print a file, just select it from the disk and click on `Send`.
- To ask for help click on the `S.O.S.` button. Note: the help provided by the staff has to be only related to issues with the computers or other physical problem.