



2025 EGOI Team Formation Test

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
T25G1	Linear Zoo	1.000 s	256 MB	4 + 8 + 15 + 14 + 18 + 16 + 25
T25G2	Sliding Crates	1.000 s	256 MB	9 + 12 + 19 + 11 + 15 + 21 + 13
T25G3	Highway Reversal	1.000 s	256 MB	14 + 16 + 22 + 18 + 19 + 11
T25G4	Grand School Award	1.000 s	256 MB	8 + 13 + 19 + 60

Notice:

Unless otherwise specified, inputs and outputs shall follow the format below:

- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.

C++ programmers should be aware that using C++ streams (`cin` / `cout`) may lead to I/O bottlenecks and substantially lower performance.

For some problems 64-bit integers may be required. In C++ it is `long long` and its token for `scanf`/`printf` is `%lld`.

All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.



T25G1 - LINEAR ZOO

Time Limit: 1.000 s / Memory Limit: 256 MB

Welcome to EGOI Zoo, where animals are kept in N display areas numbered from 1 to N , and each display area has **at most one animal**. The EGOI Zoo only displays three kinds of animals: Snakes $\boxed{S}$, Lions $\boxed{L}$, and Parrots $\boxed{P}$.

You have a record of which animal is in each display area, but unfortunately, some of these records are possibly incomplete. Formally, you have a string T of length N , for $1 \leq i \leq N$:

- $T_i = \boxed{?}$ if the content in the i -th display area is unknown.
- $T_i = \boxed{S}$, $\boxed{L}$, or $\boxed{P}$ if the i -th display area contains a Snake, a Lion, or a Parrot respectively.
- $T_i = \boxed{N}$ if the i -th display area contains no animal.

To recover the record, you have to replace each $\boxed{?}$ in T with one of $\boxed{S}$, $\boxed{L}$, $\boxed{P}$, or $\boxed{N}$.

To help you recover the record of the EGOI Zoo, M tourists have visited the EGOI Zoo over the past week. They provided you with some essential information:

- For $1 \leq i \leq M$, the i -th tourist reports: "In the display area X_i , there is **NO** animal of type Y_i ", where $1 \leq X_i \leq N$ and $Y_i = \boxed{S}$, $\boxed{L}$ or $\boxed{P}$. For example, if $(X_i, Y_i) = (3, \boxed{P})$, there is no Parrot in the 3-rd display area.

Additionally, the zookeeper reports: "Let C_S be the number of snakes, C_L be the number of lions, and C_P be the number of parrots. Then, $C_S \geq C_L \geq C_P$." For example, the recovered record cannot be $\boxed{LPP}$ regardless of the information from the tourists, as $C_S = 0$, $C_L = 1$, $C_P = 2$, $C_S \geq C_L \geq C_P$ is not satisfied.

The information from the M tourists and the zookeeper are guaranteed to be true. Using this information, can you recover the content in each of the display areas, by replacing each $\boxed{?}$ in T with one of $\boxed{S}$, $\boxed{L}$, $\boxed{P}$, or $\boxed{N}$? Any way of recovering the content in each of the display areas is accepted as long as there is no contradiction with the given information. It is guaranteed that the information is consistent, i.e. there is at least one way to recover the content in the display areas.

INPUT

The first line of the input contains two integers N and M , the number of display areas and the number of tourists.

The second line of the input contains a string T , the possibly incomplete record of which animal is in each of the display areas.

The next M lines of the input contain information from the tourists. The i -th ($1 \leq i \leq M$) of these lines comes in the form of an integer X_i and a character Y_i , describing the information from the i -th tourist.

OUTPUT

Output a possible recovered string T' , for $1 \leq i \leq N$:

- If $T_i \neq \boxed{?}$ in the input, then $T'_i = T_i$.
- If $T_i = \boxed{?}$ in the input, then T'_i is one of $\boxed{S}$, $\boxed{L}$, or $\boxed{P}$ if the i -th display area contains a Snake, Lion, or Parrot respectively, or $\boxed{N}$ if the i -th display area contains no animal.

If there are multiple solutions, output any of them.



SAMPLE TESTS

	Input	Output				
1	<table><tr><td>1 1</td></tr><tr><td>?</td></tr><tr><td>1 S</td></tr></table>	1 1	?	1 S	<table><tr><td>N</td></tr></table>	N
1 1						
?						
1 S						
N						

The only display area does not contain a Snake. To satisfy the information from the zookeeper ($C_S \geq C_L \geq C_P$), we can only set the display area to having no animal ($T_1 = \boxed{N}$). This way, $C_S = C_L = C_P = 0$.
This sample satisfies the constraints of subtask 1, 2, 6, and 7.

2	8 4	SSSSNPLN
	SSSS?P??	
	6 S	
	7 S	
	8 S	
	5 L	

In the output SSSSNPLN, the 6-th to 8-th display area does not contain S, and the 5-th display area does not contain L, which matches the information from the tourists.
The output also matches the information from the zookeeper, as $C_S = 4$, $C_L = 1$, and $C_P = 1$.
Other outputs are accepted as long as there is no contradiction with the given information. (e.g. SSSSNPLL and SSSSPPLL)
This sample satisfies the constraints of subtask 4, 6, and 7.

3	<table><tr><td>5 0</td></tr><tr><td>P??L?</td></tr></table>	5 0	P??L?	<table><tr><td>PSSLL</td></tr></table>	PSSLL
5 0					
P??L?					
PSSLL					

This sample satisfies the constraints of subtask 5, and 7.

SUBTASKS

For all cases:

$$1 \leq N \leq 200$$

$$0 \leq M \leq 600$$

$$T_i = \boxed{?}, \boxed{S}, \boxed{L}, \boxed{P} \text{ or } \boxed{N} \text{ for } 1 \leq i \leq N$$

$$1 \leq X_i \leq N \text{ for } 1 \leq i \leq M$$

$$Y_i = \boxed{S}, \boxed{L} \text{ or } \boxed{P} \text{ for } 1 \leq i \leq M$$

All (X_i, Y_i) pairs are distinct

It is guaranteed that a solution exists

	Points	Constraints
1	4	$N = 1$ $M = 1$ $(X_1, Y_1) = (1, \boxed{S})$
2	8	$T_i = \boxed{?}$ for $1 \leq i \leq N$
3	15	$T_i = \boxed{S}$ for $1 \leq i \leq \left\lceil \frac{N}{2} \right\rceil$ $M = 0$
4	14	$T_i = \boxed{S}$ for $1 \leq i \leq \left\lceil \frac{N}{2} \right\rceil$
5	18	$M = 0$
6	16	There is no area i with the possibility of being a snake and the possibility of being a lion Formally, for all i ($1 \leq i \leq N$) where $T_i = \boxed{?}$, there exists some j ($1 \leq j \leq M$), such that $X_j = i$, and Y_j is either $\boxed{S}$ or $\boxed{L}$
7	25	No additional constraints

T25G2 - SLIDING CRATES

Time Limit: 1.000 s / Memory Limit: 256 MB

In the kingdom of EGOI, there is a canyon of width N . The canyon can be represented as a two-dimensional grid of width N . We will use a coordinate system where each cell in the grid is denoted using coordinates (X, Y) , where $1 \leq X \leq N$ and $1 \leq Y \leq 10^9$. Cell $(X, 1)$ is the tallest possible cell in column X , and cells with a larger y -coordinate are deeper cells in column X . Therefore, cell $(1, 1)$ is located at the tallest position in the leftmost column. The depth of the canyon at x -coordinate i is D_i , meaning that cells from $(i, 1)$ to (i, D_i) are unoccupied, while cells $(i, D_i + 1)$ and all cells below are occupied by stone.



Figure 1: A canyon with $N = 6$ and $D = \{1, 3, 2, 3, 3, 2\}$.

Unfortunately, some visitors are trapped within the canyon. As the king of EGOI, you have decided to drop supply crates into the canyon to rescue them before further assistance arrives. The crates vary in width but always have a height of 1. A crate with width W occupies a $W \times 1$ rectangular region of cells. We denote the **position** of a crate as the column of the leftmost cell (smallest x -coordinate) that the crate occupies. That is, if a crate occupies cells (X_1, Y) to (X_2, Y) , its position will be X_1 .

You plan to drop M crates. The i -th crate has width W_i and will be dropped at position X_i . The crate will fall vertically, until it encounters a stone cell. Therefore, the crate will initially occupy cells from (X_i, h) to $(X_i + W_i - 1, h)$, where h is the minimum value among D_{X_i} to $D_{X_i + W_i - 1}$. In the example below in Figure 2, to figure out where the crate with width 4 dropped at position 2 will occupy, we look at the depths D_2, D_3, D_4, D_5 . The depths are 3, 2, 3, 3 and the minimum among all values is 2. Therefore, the crate lands at height $h = 2$, occupying cells $(2, 2)$ to $(5, 2)$.

Once the crate comes to rest, it will slide down the canyon whenever possible. Due to mysterious forces, it will only slide to the right (in the positive x direction). A crate can slide from position X to position $X + 1$ if and only if it can reach a lower position (greater y -coordinate) after sliding. More formally, a crate can slide from occupying cells (X, h) to $(X + W - 1, h)$ to occupying cells $(X + 1, h')$ to $(X + W, h')$ if and only if $h' > h$ and $X + W \leq N$, where h is the minimum value among D_X to D_{X+W-1} and h' is the minimum value among D_{X+1} to D_{X+W} . **A crate may slide multiple times to the right until it can no longer slide.**



Figure 2: Dropping a crate with $W_i = 4$, $X_i = 2$.



Figure 3: The crate will slide from cell $(1, 2)$ to cell $(2, 3)$, then further slide from cell $(2, 3)$ to cell $(3, 5)$. It cannot slide anymore afterwards.

Your task is to determine the final position of each crate, which is the position of the crate after all slides. Crates are removed immediately after they cannot slide anymore, before dropping another crate. Thus, each crate can be considered independently, and the position of one crate will not affect the sliding of another crate.

INPUT

The first line of input contains two integers, N and M , representing the width of the canyon and the number of crates dropped respectively.

The second line contains N integers, $D_1, D_2, \dots, D_N$, where D_i represents the depth of the canyon at x -coordinate i .

The next M lines each contain two integers, W_i and X_i , representing the width of the i -th crate and its initial drop position, respectively.

OUTPUT

Output M lines, where the i -th line represents the final position of the i -th crate.

SAMPLE TESTS

	Input	Output
1	64 132332 42 11 25 13	2 2 5 4

The final location of all crates are shown below.



The dropping of crate 1 is shown in Figure 2. Crate 1 will not slide at all.



Crate 2 will slide once from cell (1,1) to cell (2,3).



Crate 3 will not slide at all.



Crate 4 will slide once from cell (3,2) to cell (4,3). Note that a crate will not slide when $h' = h$.

2	54 23541 11 22 32 24	3 3 2 4
---	---	---

Refer to Figure 3 above. Crate 1 will slide 2 times before stopping.

**SUBTASKS**

For all cases:

$$1 \leq N, M \leq 10^5$$

$$1 \leq D_i \leq 10^9 \text{ for all } 1 \leq i \leq N$$

$$1 \leq W_j \leq N \text{ for all } 1 \leq j \leq M$$

$$1 \leq X_j \leq N - W_j + 1 \text{ for all } 1 \leq j \leq M$$

	Points	Constraints
1	9	$D_1 < D_2 < \dots < D_N$
2	12	$W_j = 1$ for all $1 \leq j \leq M$ $N, M \leq 500$
3	19	$W_j = 1$ for all $1 \leq j \leq M$
4	11	$N, M \leq 500$
5	15	It is guaranteed that each crate slides down at most 10 times
6	21	$D_1 < D_2 < \dots < D_K > D_{K+1} > \dots > D_N$ for some $1 < K < N$
7	13	No additional constraints



T25G3 - HIGHWAY REVERSAL

Time Limit: 1.000 s / Memory Limit: 256 MB

The kingdom of EGOI consists of N cities. To accommodate the large amount of residents travelling between cities, Alice, the queen of EGOI kingdom, has ordered the construction of M one-way highways, where the i -th highway goes from city U_i to city V_i and requires W_i minutes to travel through. No two highways start and end at the same pair of cities, and no highways start and end at the same city.

Not long after finishing the construction of the highways, Alice realizes a problem. While the traffic is smooth for the majority of a day, it gets significantly worse during the after-hours when people travel back home. To avoid building more highways, Alice proposed to reverse the direction of all highways after a predetermined *reversal time*, that is, if a highway goes from city u to city v before the reversal time, it will instead go from city v to city u starting on the reversal time while requiring the same amount of time to travel through it. Travellers are forbidden to be on any highway when the reversal happens, as it may cause confusion and pose danger to others. Specifically, any travelling done on the highways must either end at or before the reversal time, or start at or after the reversal time. A traveller can wait at a city for any amount of time, but the waiting time will also contribute towards the total travelling time.

Alice also wants to take into consideration her own travelling time when determining the reversal time. She leaves her resort at city R and travels to the capital of EGOI kingdom at city C every day at the 0-th minute. She now has Q potential candidates for the reversal time, and she would like to know the minimum time she will need to arrive at the capital for each of the Q candidate times. The path Alice takes can be different across different candidate times, as long as it is the optimal path for the given reversal time. As the wisest programmer of the kingdom, help queen Alice find the answer!

INPUT

The first line of the input contains three integers N, M, Q , the number of cities, the number of highways and the number of candidates for the reversal time.

The second line of the input contains two integers R, C , the city that Alice's resort is at and the capital city of EGOI kingdom.

The next M lines of the input each contains three integers U_i, V_i, W_i , the starting city, ending city and time required to travel for the i -th highway.

The next Q lines of the input each contains an integer T_i , the i -th candidate reversal time.

OUTPUT

For each of the Q candidate reversal times, output the minimum amount of time (in minutes) that Alice needs to travel from city R to city C . If it is impossible to travel from city R to city C , output $\boxed{-1}$.

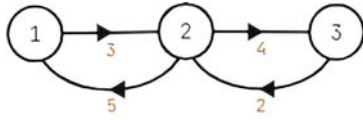
SAMPLE TESTS

	Input	Output																										
1	<table><tr><td>3</td><td>4</td><td>2</td></tr><tr><td>1</td><td>3</td><td></td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>2</td><td>1</td><td>5</td></tr><tr><td>2</td><td>3</td><td>4</td></tr><tr><td>3</td><td>2</td><td>2</td></tr><tr><td>10</td><td></td><td></td></tr><tr><td>4</td><td></td><td></td></tr></table>	3	4	2	1	3		1	2	3	2	1	5	2	3	4	3	2	2	10			4			<table><tr><td>7</td></tr><tr><td>6</td></tr></table>	7	6
3	4	2																										
1	3																											
1	2	3																										
2	1	5																										
2	3	4																										
3	2	2																										
10																												
4																												
7																												
6																												

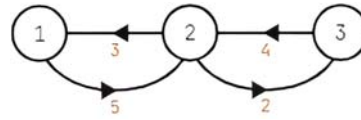
Input

Output

The highway layout of EGOI kingdom before the reversal time is as follows:



And the highway layout after the reversal time is as follows:



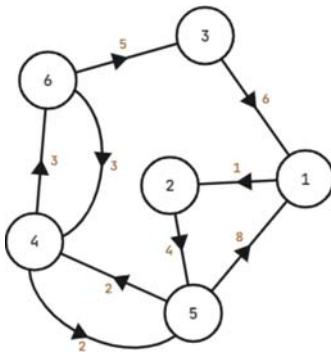
For reversal time $t = 10$, Alice can go from city 1 to city 2 in 3 minutes, and from city 2 to city 3 in 4 minutes. Hence the total travelling time is 7 minutes.

For reversal time $t = 4$, Alice can go from city 1 to city 2 in 3 minutes. She cannot get on the highway from city 2 to city 3 before the reversal time, as she will not be able to finish the journey before then. She has to wait 1 minute at city 2, after which she can go from city 2 to city 3 in 2 minutes. Therefore, the total travelling time is 6 minutes.

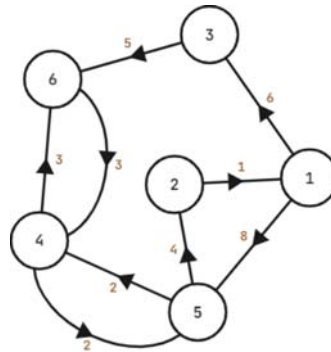
2

6 9 2	10
1 6	12
1 2 1	
3 1 6	
2 5 4	
5 1 8	
5 4 2	
4 5 2	
4 6 3	
6 4 3	
6 3 5	
13	
1	

The highway layout of EGOI kingdom before the reversal time is as follows:



And the highway layout after the reversal time is as follows:



For reversal time $t = 13$, Alice can go from city 1 to city 6 through the path $1 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$ for a total travelling time of 10 minutes.

For reversal time $t = 1$, Alice can wait at city 1 for 1 minute, and go from city 1 to city 6 through the path $1 \rightarrow 3 \rightarrow 6$ for a total travelling time of 12 minutes.

SUBTASKS

For all cases:

$$2 \leq N \leq 10^5$$

$$1 \leq M \leq 2 \times 10^5$$

$$1 \leq Q \leq 10^5$$

$$1 \leq R, C \leq N$$

$$R \neq C$$

$$1 \leq U_i, V_i \leq N \text{ for } 1 \leq i \leq M$$

$$U_i \neq V_i \text{ for } 1 \leq i \leq M$$

$$(U_i, V_i) \neq (U_j, V_j) \text{ for } 1 \leq i < j \leq M$$

$$1 \leq W_i \leq 10^9 \text{ for } 1 \leq i \leq M$$

$$1 \leq T_i \leq 10^{18} \text{ for } 1 \leq i \leq Q$$

	Points	Constraints
1	14	$Q = 1$ $M = 2N - 2$ $U_{2i-1} = i, V_{2i-1} = i + 1 \text{ for } 1 \leq i \leq N - 1$ $U_{2i} = i + 1, V_{2i} = i \text{ for } 1 \leq i \leq N - 1$
2	16	$M = 2N - 2$ $U_{2i-1} = i, V_{2i-1} = i + 1 \text{ for } 1 \leq i \leq N - 1$ $U_{2i} = i + 1, V_{2i} = i \text{ for } 1 \leq i \leq N - 1$
3	22	$Q = 1$ $T_1 = 10^{18}$
4	18	$Q = 1$ $N \leq 200$
5	19	$Q = 1$
6	11	No additional constraints

T25G4 - GRAND SCHOOL AWARD

Time Limit: 1.000 s / Memory Limit: 256 MB

In HKGOI, there are three kinds of medals: Gold, Silver and Bronze. Gold, Silver and Bronze Medals are worth 5, 2 and 1 points respectively.

There are N students. The students are assigned ID $1, 2, \dots, N$. Each of the students won exactly one medal. **It is guaranteed that each kind of medal will be awarded to at least one student.**

You are curious about what medals each student has won. However, the only thing you can do is to compare the total scores of two groups of students. The comparison works as follows:

- Select a set of students to form group A, and another set of students to form group B. No student should be in both group A and group B.
- The total score of the students in group A is compared with that of group B. Either $<$, $>$ or $=$ is returned, which means the total score of group A is lower than, higher than or equal to that of group B respectively.

Your task is to determine the medal won by each of the N students, using as few comparisons as possible.

INPUT AND OUTPUT

Your program should read input from the standard input and write output to the standard output.

Firstly, your program should read an integer T , the number of cases. Then, T cases follow.

For each test case, your program should first read an integer N , the number of medalists.

Then, repeat the following until you can determine all the medals of the N medalists.

1. Output $?$, followed by a description of the two groups. For each group $X = A$ or B , output an integer K_X ($0 \leq K_X \leq N$), the number of medalists in that group, followed by K_X integers, the IDs of the medalists in group X . Please note that the IDs should not appear more than once, or you will receive the Wrong Answer verdict.
2. Read a character c (either $<$, $>$ or $=$). If c is $>$, it means Group A obtains a higher score. If c is $<$, it means Group B obtains a higher score. Finally, if c is $=$, it means both groups have the exact same score.

Once you have decided the answer, output $!$, followed by a string s of N characters, where s_i ($1 \leq i \leq N$) is G , S or B meaning student i obtains a Gold, Silver and Bronze medal, respectively.

Your answer will be accepted if your guess for the medals of the students is all correct. Otherwise, you will receive the Wrong Answer verdict. Note that if your output format is invalid, or if your program suddenly terminates, a Wrong Answer or Time Limit Exceeded verdict may result.

Your program should move on to the next test case after you report the answer. After solving all the T cases, your program should terminate.

Important: You must flush the output after writing every line to it. C++ contestants are advised to use `fflush(stdout);`

Note that the actual grader is **not adaptive**, i.e. the medals won by each student is fixed and does not change with the comparisons made.



SAMPLE RUN

Input	Output	Explanation
2		$T = 2$.
5		Case 1: $N = 5$.
	? 1 1 1 2	Group A: Medalist 1 (Gold), Total Score = 5 Group B: Medalist 2 (Silver), Total Score = 2
>		Group A has a higher score.
	? 1 1 1 3	Group A: Medalist 1 (Gold), Total Score = 5 Group B: Medalist 3 (Gold), Total Score = 5
=		Group A and Group B have the same score.
	? 2 2 4 1 1	Group A: Medalist 2 (Silver), Medalist 4 (Silver), Total Score = $2 + 2 = 4$ Group B: Medalist 1 (Gold), Total Score = 5
<		Group B has a higher score.
	! GSGSB	The medals for the students in ascending order of their IDs are Gold, Silver, Gold, Silver, Bronze respectively.
3		Case 2: $N = 3$.
	? 1 1 1 2	Group A: Medalist 1 (Gold), Total Score = 5 Group B: Medalist 2 (Silver), Total Score = 2
>		Group A has a higher score.
	? 1 2 1 3	Group A: Medalist 2 (Silver), Total Score = 2 Group B: Medalist 3 (Bronze), Total Score = 1
>		Group A has a higher score.
	! GSB	The medals for the students in ascending order of their IDs are Gold, Silver, Bronze respectively.

The comparisons shown in this example are just illustrations of the format; they might not represent an efficient way to find the medals.

SAMPLE TESTS

	Input	Output
1	2 5 GSGSB 3 GSB	[Case 1] Accepted: 3 comparisons used [Case 2] Accepted: 2 comparisons used

This sample test corresponds to the sample run as shown above.



SUBTASKS

For all cases:

$$1 \leq T \leq 100$$

$$3 \leq N \leq 90$$

M only consists of $\boxed{G}$, $\boxed{S}$, $\boxed{B}$

Each of $\boxed{G}$, $\boxed{S}$, $\boxed{B}$ appears in M at least once

	Points	Constraints
1	8	$N = 3$
2	13	$M_1 = \boxed{G}$, $M_2 = \boxed{S}$, $M_3 = \boxed{B}$
3	19	For X in $\boxed{G}$, $\boxed{S}$ or $\boxed{B}$, $M_i = X$ for some $i \leq 7$
4	60	No additional constraints

SCORING

For a test case, you will receive zero score if your program does not exit correctly or any of your guess is incorrect. Otherwise, your score will be calculated depending on the number of comparisons you made, as follows.

In Subtask 1, 2 and 3, if you made at most 100 comparisons, you get full marks. otherwise, you get 0.

In Subtask 4, you can obtain a partial score. Suppose you make a K comparisons in a test case. Your score will be calculated according to following:

K	Score
$K \leq N$	60
$K = N + 1$	54
$K = N + 2$	48
$K = N + 3$	44
$N + 4 \leq K \leq N + 10$	40
$N + 10 < K \leq 2N$	24
$\max\{N + 10, 2N\} < K \leq 3N$	18
$\max\{N + 10, 3N\} < K \leq 1\,000$	12
$K > 1\,000$	0

Your score in a subtask is the minimum score you obtained over all the cases of that subtask.

TESTING TOOL

In order to test your program, you may download the [testing tool](#). Note that the actual grader used is different from the testing tool.

To use the sample interactor, you should create an input file, such as `sample1.in`. The format of the input file should be as follows:

The first line consists of an integer T , the number of cases. Then, T cases follows.

The format of a case is as follows:

- The first line consists of an integer N , the number of medalists.
- The next line consists of a string M of length N , where M_i is the medal obtained by student i ($M_i = \boxed{G}$, $\boxed{S}$ or $\boxed{B}$ for Gold, Silver and Bronze respectively).

For C++ programs, say `award.cpp`, you can first compile with the command

```
g++ -static -std=c++20 -Wno-unused-result -s -O2 -Ilib -o award award.cpp
```

and then run:

```
python3 testing_tool.py ./award < sample1.in
```

For each case, if your program is judged as incorrect, the sample interactor outputs a line `[Case %d] Error: Reason` and terminate immediately. Otherwise, it outputs a line `[Case %d] Accepted: %d comparisons used`. The sample interactor will interact for at most 1 000 comparisons for each case.

Please note that if an invalid input is supplied to the sample interactor, or your output during interaction is invalid, the behaviour is undefined.