D. Guessing Game

Problem Name	Guessing Game
Time Limit	4 seconds
Memory Limit	1 gigabyte

In de oude stad Lund, is er een straat met N huizen op een rij, genummerd van 0 tot $N-1$. Emma woont in één van deze huizen en haar vrienden Anna en Bertil willen uitzoeken in welk huis. Emma besluit een spel met hun te spelen, in plaats van gewoon te vertellen waar ze woont. Voordat het spel start, weten Anna en Bertil alleen het aantal huizen in de straat. Anna en Bertil mogen een positief geheel getal K kiezen en een strategie afspreken. Communicatie na die tijd is verboden.

Het spel zelf bestaat uit twee fases. In de eerste fase, kiest Emma de volgorde waarin de huizen bezocht gaan worden, zodanig dat haar huis het laatste huis is. Daarna leidt ze Anna één voor één langs de huizen in de gekozen volgorde, zonder dat ze Anna de volgorde van tevoren vertelt. Voor elk huis dat niet Emma's huis is, mag Anna een enkel geheel getal tussen 1 en K op de voordeur schrijven met stoepkrijt. Op de deur van het laatste huis dat ze bezoeken, Emma's huis, schrijft Emma zelf een enkel geheel getal tussen 1 en K.

In de tweede fase van het spel, loopt Bertil langs de straat van huis 0 naar huis $N-1$ en leest alle getallen die op de deuren staan en die geschreven zijn door Anna en Emma. Hij moet nu raden in welk huis Emma woont. Hij heeft twee kansen om dit correct te raden. Als dit lukt, winnen Anna en hij het spel. Anders wint Emma.

Kun je een strategie maken zodat Anna en Bertil gegarandeerd winnen? Jouw strategie wordt gescoord op basis van de waarde van K (hoe kleiner, hoe beter).

Implementation

Dit is een multi-run probleem, dat betekent dat je programma meerdere keren wordt uitgevoerd. De eerste keer, voert het Anna's strategie uit. Daarna wordt de strategie van Bertil uitgevoerd.

Op de eerste regel staan twee gehele getallen P en N, waar P is ofwel 1 ofwel 2 (eerste of tweede fase), en N is het aantal huizen. N **is altijd** $100\,000$**, behalve bij de sample input**.

De volgende input hangt af van de fase:

Fase 1

Print eerst het gehele getal K ($1 \leq K \leq 1\,000\,000$). Daarna moet je programma $N-1$ keer: een regel inlezen met een index i ($0 \leq i < N$) en een regel printen met een geheel getal A_i ($1 \leq A_i \leq K$), waar A_i is het nummer dat Anna schrijft op huis i. Elke index i behalve de index van Emma's huis komt precies 1 keer voor, in een volgorde die bepaalt wordt door de grader.

Fase 2

Lees een regel met N gehele getallen $A_0, A_1, \ldots, A_{N-1}$, waar A_i is het nummer dat geschreven is op deur van huis i.

Print daarna een regel met twee gehele getallen s_1 en s_2 ($0 \leq s_i < N$), de geraden indexen. s_1 en s_2 mogen gelijk zijn.

Implementation Details

Let op dat als je je programma runt in fase 2 dat je programma wordt herstart. Dit betekent dat je niet informatie kan opslaan in variabele tussen twee runs.

Flush de standaard output na iedere lijn die je print, anders krijg je Time Limit Exceeded. In Python, `print()` flushes automatically. In C++, `cout << endl;` also flushes in addition to printing a newline; if using printf, use `fflush(stdout)`.

De grader van dit probleem is misschien **adaptief**, dus hij kan zijn antwoorden soms aanpassen om te voorkomen dat heuristische oplossingen goedgekeurd worden. Dus de grader kan fase 1 trial-runnen, naar je output kijken, en dan fase 1 echt runnen, om informatie te gebruiken uit de eerste run.

Jouw programma moet deterministisch zijn, dus als je het twee keer runt met dezelfde input, moet het twee keer hetzelfde antwoord geven. Als je dus random wilt gebruiken, zorg dat je een vaste random seed gebruikt. This can be done by passing a hard-coded constant to `srand` (in C++) or `random.seed` (in Python). If using C++11's random number generators, by specifying the seed when constructing the random number generator. In particular, you can not use `srand(time(NULL))` in C++.

Als de grader erachter komt dat je programma niet deterministisch is, dan krijg je een Wrong Answer.

Als de *som* van de runtimes van de (tot 3) verschillende runs van jouw programma de tijdslimiet overschrijft, krijg je een Time Limit Exceeded.

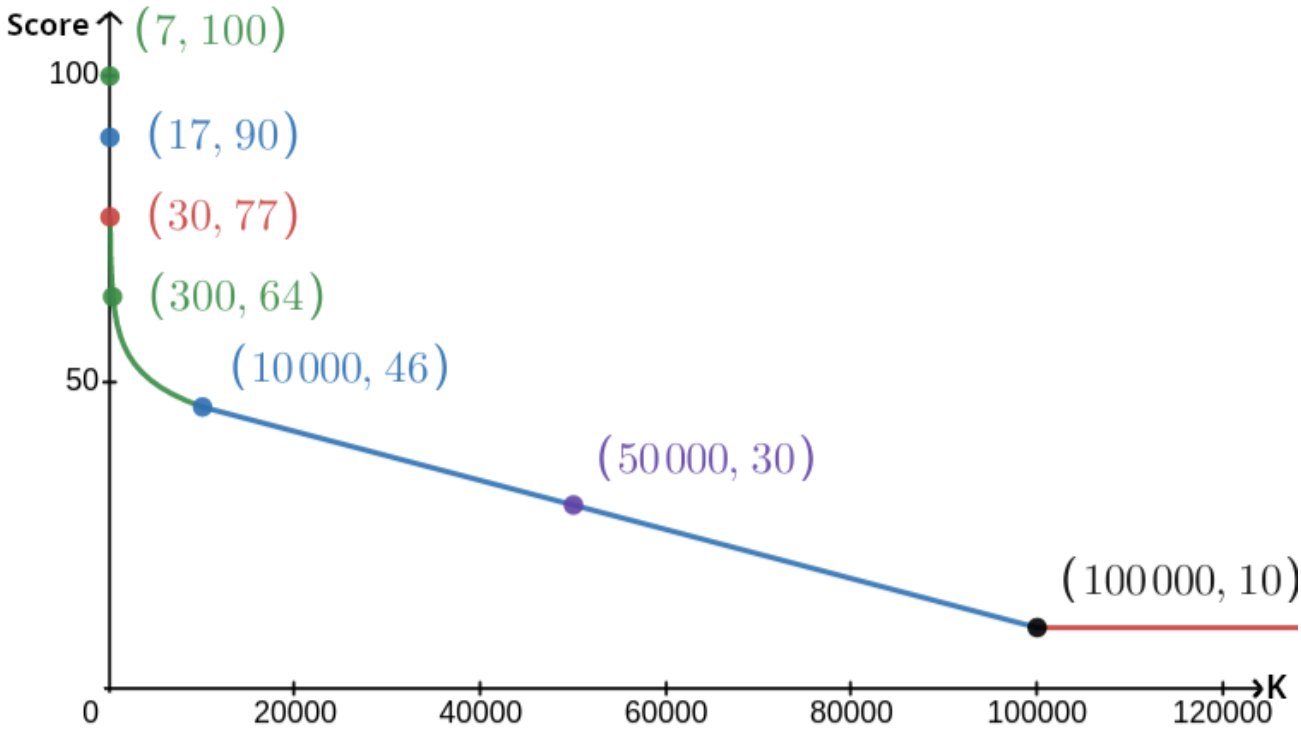
Scoring

Your solution will be tested on a number of test cases. If your solution fails on *any* of these test cases (e.g. by giving wrong answers (Wrong Answer), crashing (Run-Time Error), exceeding the time limit (Time Limit Exceeded), etc.), you will receive 0 points and the appropriate verdict.

If your program successfully finds the index of Emma's house in *all* test cases, you will get the verdict Accepted, and a score calculated as follows. Let K be the maximum value of K used for any test case. Depending on K:

	Score
$K > 99\,998$	10 points
$10\,000 < K \leq 99\,998$	$10 + \lfloor 40(1 - K/10^5) \rfloor$ points
$30 < K \leq 10\,000$	$46 + \lfloor 31(4 - \log_{10}(K))/(4 - \log_{10}(30)) \rfloor$ points
$7 < K \leq 30$	$107 - K$ points
$K \leq 7$	100 points

The scoring function is depicted in the figure below.



The sample test case is ignored for scoring, and your solution does not have to work on it.

Testing Tool

To facilitate the testing of your solution, we provide a simple tool that you can download. See "attachments" at the bottom of the Kattis problem page. The tool is optional to use, and you are allowed to change it. Note that the official grader program on Kattis is different from the testing tool.

Example usage (with $N = 4$, $s = 2$, where s is the number written on the last house visited):

For Python programs, say `solution.py` (normally run as `pypy3 solution.py`):

```
python3 testing_tool.py pypy3 solution.py <<<"4 2"
```

For C++ programs, first compile it (e.g. with `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) and then run:

```
python3 testing_tool.py ./solution.out <<<"4 2"
```

The testing tool will visit the houses in random order. To use a specific order, modify the testing tool where it says "MODIFY HERE".

Example Interaction

The sample test case is ignored for scoring, and your solution does not have to work on it.

Suppose we have $N = 4$ and that Emma lives in house 1. Let A be the list of numbers written on the houses. Initially, $A = [0, 0, 0, 0]$, where 0 means that no number is written on the corresponding house.

In the first run of your code:

$N = 4$ is given. Solution responds with $K = 3$.

A_2 is asked for. Solution responds with 3. A is now $[0, 0, 3, 0]$.

A_0 is asked for. Solution responds with 1. A is now $[1, 0, 3, 0]$.

A_3 is asked for. Solution responds with 2. A is now $[1, 0, 3, 2]$.

Finally, the grader sets $A_1 = 2$, so that A = $[1, 2, 3, 2]$ in the end. This marks the end of the first phase.

In the Phase 2 of your code, your solution is passed the list `1 2 3 2`.

It responds with `1 3`.

Since one of the guesses is the correct index of the house (1), Anna and Bertil win the game.

grader output	your output
1 4	
	3
2	
	3
0	
	1
3	
	2

grader output	your output
2 4	
1 2 3 2	
	1 3