# D. Guessing Game

| Naloga | Guessing Game |
|---|---|
| Omejitev časa | 4 sekunde |
| Omejitev spomina | 1 gigabyte |

V starem delu Lunda je ulica z $N$ hišami v vrsti. Označene so s števili od $0$ do $N-1$. Emma živi v eni izmed teh hiš, njena prijatelja Anna in Bertil pa želita ugotoviti v kateri. Namesto da bi Emma povedala prijateljema, kje živi, se je odločila, da se bodo igrali igro. Preden se igra začne, Anna in Bertil poznata samo število hiš v ulici. Na tej točki lahko Anna in Bertil izbereta pozitivno celo število $K$ in se dogovorita za strategijo. Po tem je komunikacija med njima prepovedana.

Igra poteka v dveh fazah. V prvi fazi Emma izbere vrstni red, v katerih bodo hiše obiskane, tako da bo njena hiša obiskana zadnja. Emma potem vodi Anno do hiš eno za drugo v tem vrstnem redu. Anna vrstnega reda ne pozna vnaprej. Na vrata vsake hiše, ki ni Emmina, lahko Emma s kredo napiše eno samo celo število med $1$ in $K$. Nazadnje obiščeta Emmino hišo, kjer Emma sama na vrata napiše eno samo število med $1$ in $K$.

V drugi fazi igre Bertil hodi po ulici od hiše $0$ do $N-1$ in bere vsa števila, ki sta jih na vrata zapisali Anna in Emma. Uganiti želi, v kateri izmed hiš živi Anna. Ima dve možnosti, da ugane pravilno in Bertil in Anna zmagata v igri. Sicer zmaga Emma.

Ali lahko najdeš strategijo, kjer Anna in Bertil zagotovo zmagata? Tvoja strategija bo ocenjena glede na vrednost $K$ (manjši $K$ pomeni boljša rešitev).

## Implementacija

To je naloga, ki zahteva več zagonov tvojega programa - ko bo zagnan prvič, naj izvede Annino fazo. Drugič bo izvedel Bertilovo fazo.

V prvi vrstici vhoda sta dve celi števili $P$ in $N$, kjer je $P$ ali $1$ ali $2$ (prva (Anna) ali druga (Bertil) faza), in $N$ število hiš v ulici. **Razen v primeru (ni uporabljen za točkovanje) bo $N$ vedno enak $100\,000$.**

Naprej je vhod odvisen od faze:

### Faza 1

Tvoj program začne z izpisom števila $K$ v eni sami vrstici $(1 \leq K \leq 1\,000\,000)$. Potem naj $(N-1)$-krat ponovi sledeče: prebere naj vrstico, v kateri je indeks $i$ $(0 \leq i < N)$ in izpiše vrstico, v kateri je število $A_i$ $(1 \leq A_i \leq K)$, kjer je $A_i$ število, ki ga je Anna zapisala na vrata hiše $i$. Vsak indeks $i$ razen indeksa Emmine hiše se bo pojavil natančno enkrat v vrstnem redu, ki ga določi ocenjevalnik.

### Faza 2

Tvoj program naj prebere vrstico z $N$ celimi števili, $A_0, A_1, \ldots, A_{N-1}$, kjer je $A_i$ število zapisano na vratih hiše $i$.

Potem naj izpiše vrstico z dvema celima številoma $s_1$ in $s_2$ $(0 \leq s_i < N)$, ki predstavljata ugibana indeksa. Števili $s_1$ in $s_2$ sta lahko enaki.

### Detajli implementacije

Ko se progam zažene v fazi 2, je ponovno zagnan. To pomeni, da ne moreš shraniti informacij o nekaterih spremenljivkah med zagonoma.

Poskrbi, da izplakneš (flush) standardni izhod po zastavljanju vprašanja, sicer bo tvoj program morda ocenjen s prekoračenim časom (Time Limit Exceeded). V Pythonu `print()` izplakuje avtomatično. V C++, `cout << endl;` tudi izplakne skupaj z izpisom nove vrstice, če pa uporabljaš printf, uporabi `fflush(stdout)`.

Ocenjevalnik za to nalogo je lahko **prilagodljiv**, to pomeni, da lahko spremeni svoje obnašanje glede na izhod tvojega programa, da prepreči, da bi bile sprejete hevristične rešitve. Lahko naredi poskusni zagon faze 1, pogleda tvoj izhod, in potem zažene fazo 1 zares z informacijami, ki jih je pridobil v prejšnjem zagonu.

**Tvoj program se *mora* obnašati deterministično**, to pomeni, da se obnaša enako, če je dvakrat zagnan na istem vhodu. Če želiš v svojem programu uporabljati naključnost, nastavi seme generatorja naključnih števil. To lahko narediš z uporabo konstante v `srand` (v C++) ali `random.seed` (v Pythonu). Če uporabljaš generator naključnih števil v C++11, seme navedeš ob konstrukciji generatorja. Ne uporabljaj `srand(time(NULL))` v C++. Če ocenjevalnik zazna, da tvoj program ni deterministične, bo rešitev prejela oceno Wrong Answer.

Če *vsota* časov tekov največ $3$ ločenih zagonov tvojega programa presega omejitev časa, bo tvoja rešitev ocenenjena kot Time Limit Exceeded.

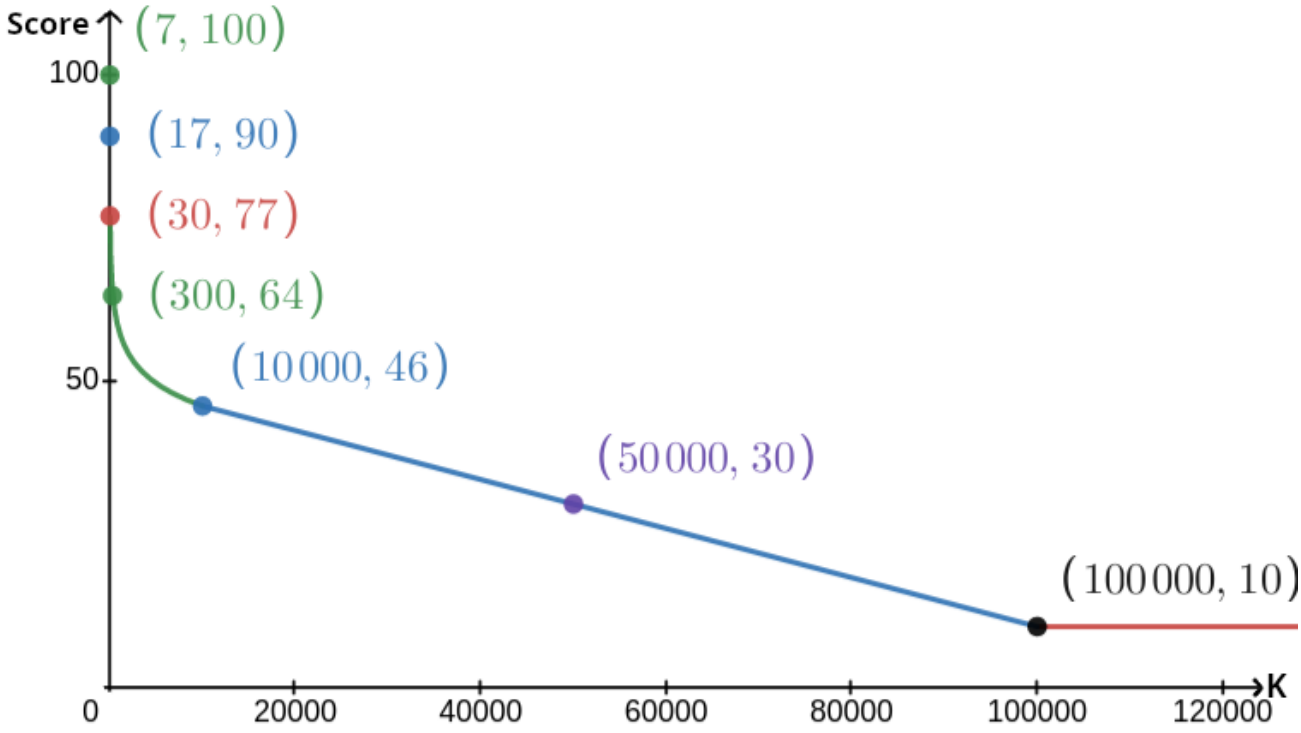
## Ocenjevanje

Tvoja rešitev bo testirana na več testnih primerih. Če tvoja rešitev pade na *kateremkoli* od teh testov (na primer, z napačnim odgovorom (Wrong Answer), sesutjem (Run-Time Error), prekoračenim časom (Time Limit Exceeded), itd.), bo prejela $0$ točk.

Če tvoj program v *vseh* primerih uspešno najde indeks Emmine hiše, bo tvoja rešitev sprejeta, ocena pa bo izračunana sledeče: Naj bo $K$ največje vrednost kateregakoli $K$ v testih. Glede na $K$

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

Funkcija, uporabljena za ocenjevanje, je prikazana na spodnji sliki.




Vzorčni testni primer (sample test case) ni uporabljen za točkovanje in ni potrebno, da tvoja rešitev deluje na vzorcu.

## Orodje za testiranje

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.

Da ti omogočimo testiranje rešitve, ti ponujamo preprosto orodje za testiranje, ki si ga lahko preneseš. Poglej med priponke (“attachments”) na dnu kattis strani z nalogo. Orodje uporabljaj po želji in lahko ga spremeniš. Uradni ocenjevalnik na kattis je drugačen od orodja za testiranje.

Primer uporabe (za $N = 4$, $s = 2$, kjer je $s$ število zapisano na zadnji hiši):

Za python program z imenom `solution.py` (običajno ga poženeš z `pypy3 solution.py`):

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

C++ programe najprej prevedi (na primer `g++ -std=gnu++17 solution.cpp -o solution.out`) in potem poženi:

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

Orodje za testiranje hiše obiskuje v naključnem vrstnem redu. Če želiš uporabiti specifičen vrstni red, spremeni orodje za testiranje na mestu, kjer piše “MODIFY HERE”.

## Primer interakcije

Vzorčni testni primer (sample test case) ni uporabljen za točkovanje in ni potrebno, da tvoja rešitev deluje na vzorcu.

Recimo, da je $N = 4$ in da Emma živi v hiši $1$. Naj bo $A$ seznam števil zapisanih na vratih. Na začetku je $A = [0, 0, 0, 0]$, kjer $0$ pomeni, da na vratih ni zapisane številke.

V prvem zagonu:

Podan je $N = 4$. Tvoja rešitev se odzove s $K = 3$.

Zahtevan je $A_2$ in tvoja rešitev se odzove s $3$. Tako je $A$ $[0, 0, 3, 0]$.

Zahtevan je $A_0$ in tvoja rešitev se odzove z $1$. Tako je $A$ $[1, 0, 3, 0]$.

Zahtevan je $A_3$ in tvoja rešitev se odzove z $2$. Tako je $A$ $[1, 0, 3, 2]$.

Na koncu ocenjevalnik nastavi $A_1 = 2$. Tako je na koncu A = $[1, 2, 3, 2]$. Tu se prva faza konča.

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

V drugem zagonu:

V drugi fazi, tvoja rešitev prejme seznam `1 2 3 2`.

Odzove se z `1 3`.

Eno od ugibanih števil je pravilno - indeks hiše (1), in Anna in Bertil zmagata v igri.

| izhod ocenjevalnika | tvoj izhod |
|---|---|
| 1 4 | |
| | 3 |
| 2 | |
| | 3 |
| 0 | |
| | 1 |
| 3 | |
| | 2 |

| izhod ocenjevalnika | tvoj izhod |
|---|---|
| 2 4 | |
| 1 2 3 2 | |
| | 1 3 |