

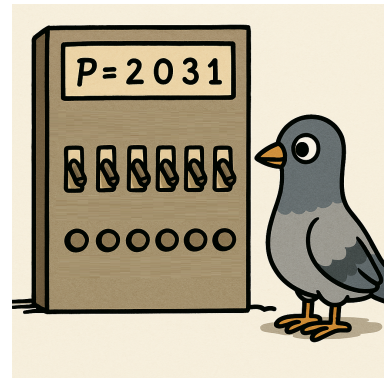


PERMUTÁCIÓ

Gimi, a galamb, elég nagy kalamajkába keveredett!

Talált egy régi gépet. A gép belsejében N titkos kapcsoló van: $s[0], s[1], s[2], \dots, s[N-1]$. Bármely i kapcsoló lehet ki-kapcsolt állapotban (amit $s[i] = 0$ jelöl), vagy bekapcsolt állapotban (amit $s[i] = 1$ jelöl). Kezdetben az összes kapcsoló ki van kapcsolva. A géphez tartozik továbbá N darab, 0 -tól $N-1$ -ig számozott gomb, egy kijelző, valamint egy titkos permutáció: $P = p[0], p[1], p[2], \dots, p[N-1]$. A permutáció a $0, 1, 2, \dots, N-1$ számok permutációja.

A gépen a következő felirat olvasható: „Ha ki tudod találni a P permutációt, nagy nyereményben lesz részed!” Gimi természetesen szeretné a nyereményt, ezért hozzád fordul segítségért.



Nagy kalamajka

A P permutáció kitalálásához megnyomhatod a gép gombjait. Amikor megnyomod az i -edik gombot, a következő történik:

- Legyen k az összes korábbi gombnyomás száma, a mostanit **nem** beleértve.
- A kapcsoló állapota a $p[k \% N]$ indexen megváltozik (formálisan: $s[p[k \% N]] := 1 - s[p[k \% N]]$);
- A kijelzőn megjelenik az i indexű kapcsoló aktuális állapota ($s[i]$ értéke).

FELADAT Adott az *Negész* szám. Ki kell találnod a titkos permutációt a gép gombjainak megfelelő számú megnyomásával.

Mivel a gép régi, tönkremegy ha túl sokszor nyomják meg a gombokat, ezért ha a gombnyomások száma meghaladja az $50 \cdot N$ értéket, nem kapod meg a nyereményt.

Ugyanezen okokból szeretnéd minimalizálni a gombnyomások számát. Erről a Pontozás fül alatt találsz a részleteket.

IMPLEMENTÁCIÓ A beadott fájlban tartalmaznia kell a következő sort:

```
#include "permutation.h".
```

A következő függvényt kell implementálnod:

```
std::vector<int> solve(int N);
```

Ez a függvény megkapja paraméterként N értékét, és egy N méretű q vektort kell visszaadnia, amelyre minden $0 \leq i \leq N-1$ esetén teljesül, hogy $q[i] = p[i]$. Ezt a függvényt tesztésenként pontosan egyszer hívja meg az értékelő.

A megoldásodban meghívhatod a következő függvényt:

```
int press_button(int i);
```

Ez a függvény visszaadja az i -edik gomb megnyomásának eredményét, vagyis az $s[i]$ aktuális értékét a fenti 2. és 3. lépés után.



- KORLÁTOK** ◆ $N \leq 10^4$
◆ P a $0, 1, 2, \dots, N-1$ számok egy permutációja.

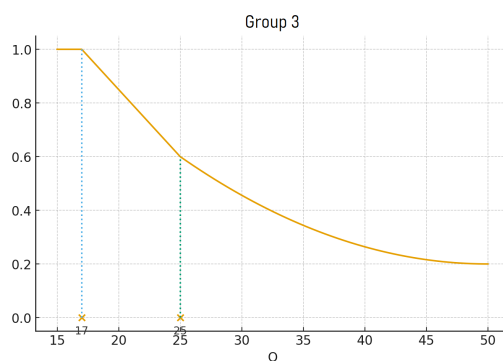
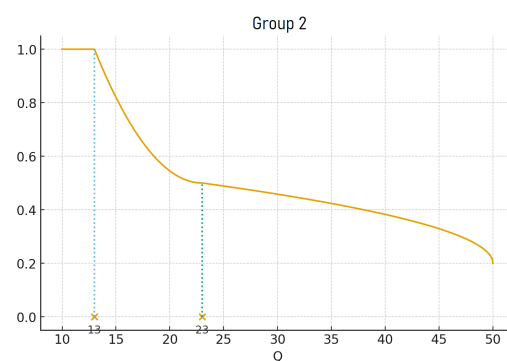
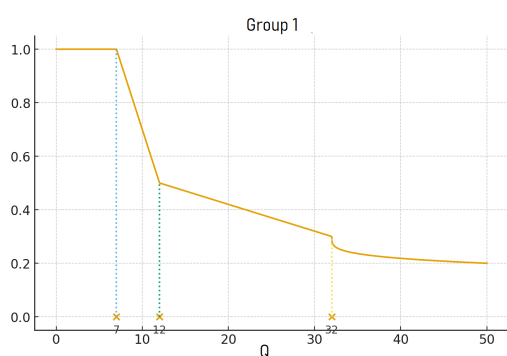
#	Pontozás	Korlátok
1	20	$N = 32$
2	40	$N = 1\,000$
3	40	$N = 10\,000$

PONTOZÁS Minden teszteset pontozása a következőképpen történik: Legyen K az a szám, ahányszor a megoldásod meghívta a `press_button()` függvényt. Legyen továbbá $Q = \frac{K}{N}$. Ha a permutáció kitalálása hibás, vagy ha $Q > 50$, akkor a pontszám 0. Egyébként a pontszám S szorozva a csoport maximális pontértékével, ahol S az alábbiak szerint definiált:

$$\text{Group 1. } S = \begin{cases} 1, & Q \leq 7 \\ 1 - \frac{1}{10}(Q - 7), & 7 < Q \leq 12 \\ \frac{1}{2} - \frac{1}{100}(Q - 12), & 12 < Q \leq 32 \\ \frac{3}{10} - \frac{1}{10} \sqrt[4]{\frac{Q-32}{18}}, & 32 < Q \leq 50 \end{cases}$$

$$\text{Group 2. } S = \begin{cases} 1, & Q \leq 13 \\ \frac{1}{2} + \frac{1}{2} \left(\frac{23-Q}{10} \right)^2, & 13 < Q \leq 23 \\ \frac{1}{5} + \frac{3}{10} \sqrt{\frac{50-Q}{27}}, & 23 < Q \leq 50 \end{cases}$$

$$\text{Group 3. } S = \begin{cases} 1, & Q \leq 17 \\ \frac{3}{5} + \frac{25-Q}{20}, & 17 < Q \leq 25 \\ \frac{1}{5} + \frac{2}{5} \left(\frac{50-Q}{25} \right)^2, & 25 < Q \leq 50 \end{cases}$$





■ **INTERAKCIÓ PÉLDA** Tekintsünk egy tesztesetet $N = 4$ és $P = [1, 3, 0, 2]$ értékekkel, valamint a következő `press_button()` hívásokkal.

Action	S	Return Value
<code>press_button(0)</code>	0100	
		0
<code>press_button(1)</code>	0101	
		1
<code>press_button(2)</code>	1101	
		0
<code>press_button(3)</code>	1111	
		1
<code>press_button(1)</code>	1011	
		0
<code>press_button(3)</code>	1010	
		0
<code>press_button(2)</code>	0010	
		1
<code>return {1,3,0,2}</code>		

■ **SAMPLE GRADER** A feladathoz egy példa kiértékelő program is tartozik (`sample-grader.cpp`). A grader a következő formátumban olvassa be a bemenetet:

- line 1: N (egész szám)
- line 2: $p[0], p[1], p[2], \dots, p[N-1]$ (N darab egész szám).

Ezután meghívja a `solve(N)` függvényt, és kiírja a megoldásod eredményét, valamint a lekérdezések számát vagy a hiba magyarázatát.