

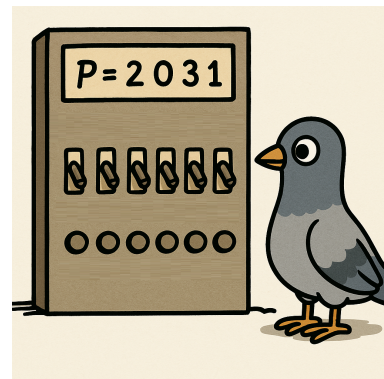


ZADANIE AUTOMAT

Adam znalazł się w ciekawej sytuacji.

Dobrał się do starego automatu, w którego środku jest N ukrytych przełączników $s[0], s[1], s[2], \dots, s[N-1]$. Każdy przełącznik może być zgaszony (wtedy $s[i] = 0$) albo zapalony (wtedy $s[i] = 1$). Początkowo wszystkie przełączniki są zgaszone. Automat ma również N przycisków ponumerowanych od 0 do $N-1$, wyświetlacz, i ukrytą permutację $P = p[0], p[1], p[2], \dots, p[N-1]$, liczb $0, 1, 2, \dots, N-1$.

Baner na automacie mówi: “jeśli zgadniesz P , wygrasz dużą nagrodę!”, Adam chciałby wygrać nagrodę ale jest zajęty graniem w Clash Royale, więc poprosił Ciebie o pomoc.



Ciekawa sytuacja

Żeby odgadnąć P , możesz klikać przyciski automatu. Kiedy przyciśniesz i -ty przycisk, zajdzie:

1. Niech k będzie liczbą twoich kliknięć poprzedzających obecnie.
2. Przełącznik o indeksie $p[k \% N]$ zostaje odwrócony (konkretniej, $s[p[k \% N]] := 1 - s[p[k \% N]]$);
3. Wyświetlacz pokaże stan przełącznika i (wartość $s[i]$).

ZADANIE Dana jest liczba N . Należy zgadnąć P , klikając przyciski możliwie niedużo razy.

Ponieważ automat jest już nadgryziony zębem czasu, rozpadnie się całkowicie jeśli klikniesz więcej niż $50 \cdot N$ przycisków, nie dostaniesz wtedy nagrody.

Z tego samego powodu, chcesz zminimalizować liczbę kliknięć. Dokładniejsze ograniczenia znajdują się w sekcji Ocenianie.

IMPLEMENTACJA Wysłany plik musi zawierać `#include "permutation.h"`.

Należy zaimplementować następującą funkcję:

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

Ta funkcja dostaje N jako argument i musi zwrócić vector q rozmiaru N taki, że dla każdego $0 \leq i \leq N-1$, $q[i] = p[i]$. Ta funkcja będzie wywoływana tylko raz w jednym teście.

W twoim rozwiązaniu możesz wywołać funkcję

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

zwróci ona efekt przyciśnięcia i -tego przycisku, to jest wartość $s[i]$, po wykonaniu kroków 2 i 3 opisanych powyżej.



- OGRANICZENIA** ◆ $N \leq 10^4$
◆ P jest permutacją liczb $0, 1, 2, \dots, N - 1$.

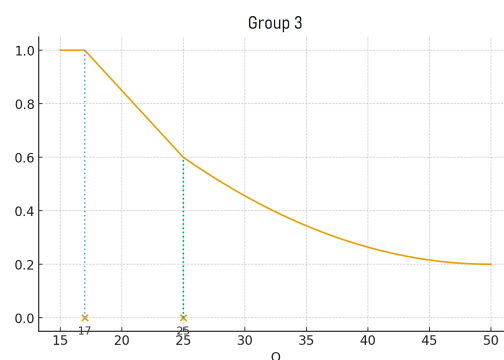
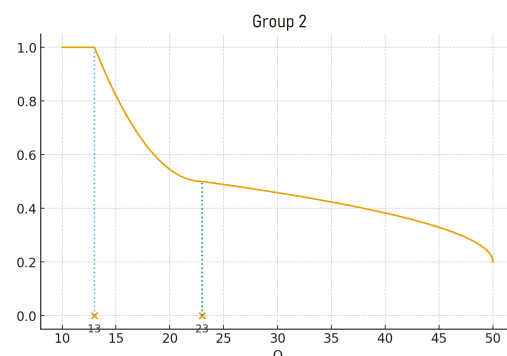
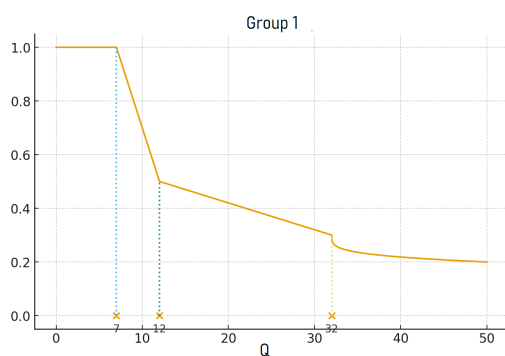
#	Punkty	Ograniczenia
1	20	$N = 32$
2	40	$N = 1\,000$
3	40	$N = 10\,000$

OCENIANIE Każdy test będzie oceniany w następujący sposób: Niech K będzie liczbą razy `press_button()` została wywołana przez Twoje rozwiązanie. Niech $Q = \frac{K}{N}$. Jeśli zwrócona przez Ciebie permutacja jest niepoprawna lub $Q > 50$, to twój wynik to 0. W przeciwnym razie, twój wynik to S pomnożone przez maksymalny możliwy wynik do uzyskania w danym podzadaniu, gdzie S jest zdefiniowane następująco:

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

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

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





■ **PRZYKŁADOWA INTERAKCJA** Rozważmy test, gdzie $N = 4$ i $P = [1, 3, 0, 2]$, z następującymi wywołaniami `press_button()`.

Akcja	S	Zwrócona wartość
<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>		

■ **PRZYKŁADOWA GRADERKA** Masz dostęp do przykładowej graderki (`sample-grader.cpp`). Graderka przyjmuje wejście w następującym formacie:

- linia 1: liczba N
- linia 2: N liczb $p[0], p[1], p[2], \dots, p[N-1]$.

Następnie graderka wywołuje funkcję `solve(N)` i wypisuje werdykt dla wyniku podanego przez Twój program razem z liczbą zadanych pytań lub wyjaśnieniem błędu.