Odzyskiwanie permutacji **(odzyskiwanie-permutacji)**

Limit pamięci: 64 MB Limit czasu: 2.50 s

W tym zadaniu celem jest odgadnąć wzór tajemniczej funkcji π.
O funkcji π wiadomo, że:

- przyjmuje tylko jeden parametr x: liczbę naturalną od 1 do N włącznie,
- zwraca jako wynik liczbę naturalną od 1 do N włącznie,
- dla każdych dwóch różnych parametrów, funkcja zwraca różne wartości (dla każdego $1 \leq x_1 \neq x_2 \leq N$ zachodzi $\pi(x_1) \neq \pi(x_2)$),
- dla każdej liczby naturalnej y od 1 do N włącznie, istnieje parametr x, że funkcja zwraca wynik y dla tego parametru (dla każdego $1 \leq y \leq N$ istnieje $1 \leq x \leq N$, że $\pi(x) = y$).

Innymi słowy, funkcja π jest permutacją zbioru $\{1, 2, \ldots, N\}$.

Otrzymujesz na wejściu liczbę naturalną N, wartości k oraz ℓ oraz dla każdej liczby naturalnej od 1 do N włącznie otrzymujesz dwie wartości: $\pi^k(x)$ oraz $\pi^\ell(x)$. Dla przypomnienia: $f^n(x) = \underbrace{f(f(f(\ldots(f}_{n}(x))\ldots)))$.

Twoim celem jest obliczyć dla każdej liczby naturalnej x od 1 do N włącznie wartość $\pi(x)$, czyli ustalić jawny wzór permutacji π.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N. W drugim wierszu wejścia znajdują się dwie liczby naturalne k oraz ℓ oddzielone pojedynczym odstępem. W kolejnych N wierszach znajdują się pary liczb naturalnych oddzielone pojedynczym odstępem. Liczby zapisane w $(x+2)$-gim wierszu określają kolejno wartości $\pi^k(x)$ oraz $\pi^\ell(x)$.

Możesz założyć, że dane są tak dobrane, że rozwiązanie zadania istnieje.

Wyjście

Twój program powinien wypisać na wyjście ciąg N liczb naturalnych pooddzielanych pojedynczymi odstępami: x-ta liczba ma określać wartość $\pi(x)$. Jeżeli istnieje wiele możliwych rozwiązań, Twój program może wypisać dowolne z nich.

Ograniczenia

$1 \leq N \leq 500\,000$, $2 \leq k, \ell \leq 10^9$.

Przykład

Wejście	Wyjście
9 2 3 4 2 1 5 6 9 8 1 2 8 3 7 9 3 5 4 7 6	5 8 7 2 4 9 6 1 3

Powtarzające się ciągi **(powtarzajace-ciagi)**

Limit pamięci: 256 MB Limit czasu: 2.00 s

W Bajtocji niedługo wybory. Bajtek ma już dość słuchania przemówień, w których politycy ciągle powtarzają nazwiska przeciwników. Ostatnie przemówienie przedstawiciela Bajtockiej Zjednoczonej Partii Informatyków. podniosło mu ciśnienie: polityk ten w zasadzie ciągle powtarzał to samo. Bajtek zapisał sobie transkrypt przemówienia i chciałby policzyć ile jest w nim powtórzeń. Dokładniej, interesuje go liczba **różnych** spójnych podsłów przemówienia, które występują w nim **co najmniej trzy razy**. Pomożesz?

Napisz program, który: wczyta transkrypt przemówienia, wyznaczy liczbę różnych spójnych jego fragmentów, które występują w przemówieniu co najmniej trzykrotnie i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się niepusty ciąg małych liter alfabetu angielskiego (bez żadnych odstępów) – treść przemówienia polityka.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba całkowita – liczba różnych spójnych fragmentów przemówienia, które wystąpiły w nim co najmniej trzykrotnie.

Ograniczenia

Długość przemówienia nie przekracza $500\,000$ znaków.

Przykład

Wejście	**Wyjście**	**Wyjaśnienie**
`abaabab`	3	W tym przypadku powtarzające się fragmenty przemówienia to: `a`, `b` oraz `ab`. Fragment `a` powtarza się nawet cztery razy.

Hackowanie generatora liczb pseudolosowych (rng-hack)

Limit pamięci: 8 MB Limit czasu: 0.25 s

Jasio wymyślił przepiękny generator liczb pseudolosowych zdolny wylosować prawie idealnie losowy ciąg dużych liczb.

Poniżej znajduje się implementacja Jasia:

```
long long johnny_next_random(long long previous) {
  return (previous + (previous >> 10) + 9876543210LL) % (1LL << 60);
}

vector<long long> johnny_random_numbers(long long sequence_length) {
  const long long seed = 12345678987654321LL;
  vector<long long> sequence;
  long long value = seed;
  for (int i = 0; i < sequence_length; i++) {
    value = johnny_next_random(value);
    sequence.push_back(value);
  }
  return sequence;
}
```

Zapewne widzisz od razu, że to beznadziejna implementacja i nie powinna być użyta do zastosowań kryptograficznych (na przykład do generowania kluczy). Aby przekonać o tym Jasia, powiesz mu ile jest liczb podzielnych przez 17 w ciągu `johnny_random_numbers(N)`. Te same liczby występujące na różnych pozycjach w ciągu liczymy wielokrotnie.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna N – parametr funkcji generującej ciąg liczb pseudolosowych.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna nieujemna liczba całkowita – liczba wystąpień liczb podzielnych przez 17 w ciągu `johnny_random_numbers(N)`.

Ograniczenia

$1 \leq N \leq 10^{18}$.

Przykład

Wejście	Wyjście	Wyjaśnienie
50	5	Indeksując ciąg od 0, wartość podzielna przez 17 występuje na pozycjach: 8, 18, 29, 33 oraz 34.