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.