Dzielniki silni **(dzielniki-silni)**

Limit pamięci: 128 MB Limit czasu: 1.50 s

O autorach niektórych zadań algorytmicznych mówi się czasami, że *Mają rozmach*, jak to mawiał Stefan Siara Siarzewski, w znanym filmie komediowym. Autorzy zadań na ten sparing chcieliby, żeby tak właśnie Wam chciało się o nich powiedzieć.

Dlatego poniosła ich fantazja i wyobrazili sobie liczbę $N!$ (N silnia, czyli iloczyn kolejnych liczb naturalnych od 1 do N włącznie), dla bardzo dużej wartości N, powiedzmy że nawet do miliarda. Wyobrażacie to sobie? Dla jasności: takich liczb nie wyobraża sobie nawet interpreter Pythona, bo (przynajmniej na typowej maszynie) zabraknie mu czasu i pamięci, żeby to obliczyć.

Ale autorzy tego zadania idą dalej i wyobrażają sobie zbiór dzielników liczby $N!$. Duży co? Były już zadania na policzenie ile cyfr ma liczba elementów tego zbioru, ale tym razem chcemy policzyć co do jednego elementu ile jest dzielników nie przekraczających jakiejś niedużej liczby, powiedzmy nawet konkretnie wprost, że chodzi nam o dzielniki nie przekraczające miliarda.

Żeby nie było zbyt łatwo, limity czasu i pamięci są dla odmiany raczej niezbyt duże. Kłaniamy się nisko i pozdrawiamy z całego serca wszystkich rozwiązujących to zadanie życząc jednocześnie powodzenia w zdobyciu upragnionego akcepta.

Napisz program, który: wczyta liczbę naturalną N, wyznaczy liczbę dodatnich dzielników liczby $N!$ nie przekraczających 10^9 i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna N.

Wyjście

W pierwszym wierszu wyjścia powinna się znaleźć liczba naturalna – liczba dzielników liczby $N!$ nie przekraczających 10^9.

Ograniczenia

$1 \leq N \leq 10^9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
5	16	Liczba $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$ ma 16 dzielników: $1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120$, wszystkie są sporo mniejsze od miliarda, a więc wszystkie należy policzyć.

Kontrprzykład (kontrprzyklad)

Limit pamięci: 128 MB Limit czasu: 1.00 s

Podciągiem słowa A nazwiemy każde słowo, które możemy uzyskać usuwając niektóre litery z A i odczytując pozostałe nie zmieniając ich kolejności. Dla przykładu, `bef` jest podciągiem słowa `abcdefg`, zaś `gf` nie jest.

Jasio na lekcji informatyki dowiedział się jak rozwiązać problem znalezienia najdłuższego wspólnego podciągu dwóch słów, czyli najdłuższego słowa, które jest podciągiem obu danych słów. Po powrocie do domu chłopak bez chwili zastanowienia zaczął rozwiązywać zadanie domowe, polegające na zaimplementowaniu poznanego algorytmu. Udało mu się napisać poprawny kod, chociaż, co później wypomniała mu nauczycielka, nieoptymalny czasowo [1].

```
string dp[N+1][N+1];
string lcs(string a, string b) {
    for (int i = 1; i <= a.size(); ++i) {
        for (int j = 1; j <= b.size(); ++j) {
            if (a[i-1] == b[j-1]) dp[i][j] = dp[i-1][j-1] + a[i-1];
            else if (dp[i-1][j].size() >= dp[i][j-1].size()) dp[i][j] = dp[i-1][j];
            else dp[i][j] = dp[i][j-1];
        }
    }
    return dp[a.size()][b.size()];
}
```

Na liście zadań znajdowało się również zadanie bonusowe, które polegało na zaimplementowaniu algorytmu znajdującego najdłuższy wspólny podciąg trzech ciągów. Jasio stwierdził, że to nic trudnego – w końcu może wykorzystać swoją poprzednią implementację i najpierw znaleźć najdłuższy wspólny podciąg dwóch pierwszych słów, a następnie najdłuższy wspólny podciąg znalezionego słowa oraz słowa trzeciego.

```
string lcs3(string a, string b, string c) {
    return lcs(lcs(a, b), c);
}
```

Testy w tym zadaniu były dość słabe i okazało się, że (oczywiście błędne) rozwiązanie Jasia przeszło wszystkie testy. Chłopak chwali się, że jest mistrzem algorytmów tekstowych. Jako osoba z klasy Jasia, która poprawnie rozwiązała zadanie bonusowe, czujesz, że trzeba go sprowadzić na ziemię.

Napisz program, który dla danych liczb N, L, J, wyznaczy trzy słowa o długości N, składające się z małych liter alfabetu angielskiego, których najdłuższy wspólny podciąg ma L liter, a program Jasia zwróci słowo o długości J, lub stwierdź, że nie istnieją takie trzy słowa.

Wejście

W pierwszym (jedynym) wierszu wejścia znajdują się trzy liczby całkowite N, L, J, pooddzielane pojedynczymi odstępami.

Wyjście

Jeżeli istnieje kontrprzykład zgodny z treścią zadania, wypisz trzy słowa w trzech wierszach. Kolejność ma znaczenie! W przeciwnym wypadku wypisz jeden wiersz zawierający słowo `ACCEPTED`.

Jeżeli istnieje wiele możliwych poprawnych odpowiedzi, Twój program może wypisać dowolną z~nich.

[1] Implementacja Jasia działa w złożoności $O(n^3)$ zamiast $O(n^2)$ ze względu na kopiowanie słów. Dla odpowiednio dobranych danych może się zdarzyć, że tablica `dp` będzie wypełniona $\Omega(n^2)$ słowami o długości $\Omega(n)$.

Ograniczenia

$1 \leq N \leq 100, 0 \leq J \leq L \leq N$.

Przykład

Wejście
```
10 5 3
```

Wyjście
```
eckoyzleak
kozalkoeox
kixdoezakw
```

Wyjaśnienie
Program Jasia obliczy, że najdłuższy podciąg dwóch pierwszych słów to `kozle`, a następnie, że najdłuższy wspólny podciąg `kozle` i trzeciego słowa to `koz`. Najdłuższy wspólny podciąg ma 5 liter, na przykład `kozak`.

Wejście
```
10 10 0
```

Wyjście
```
ACCEPTED
```

Pokrycie ciągu ciągami **(pokrycie-ciagami)**

Limit pamięci: 128 MB Limit czasu: 3.00 s

Jasio, nudząc się na lekcji matematyki, wymyślił następujące zadanie algorytmiczne.

"Dla danego zbioru liczb, ile co najmniej ciągów arytmetycznych o różnicy niemniejszej niż 10 jest potrzebnych, żeby każdy element zbioru występował w którymś z ciągów. Ciągi mogą być dowolnie długie."

Niestety nie potrafi go rozwiązać. Napisz program, który rozwiąże zadanie Jasia.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N, określająca moc zbioru, który należy pokryć. W drugim wierszu wejścia znajduje się rosnący ciąg N liczb naturalnych A_i, pooddzielanych pojedynczymi odstępami. Są to elementy zbioru.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba naturalna – minimalna liczba ciągów arytmetycznych o różnicy co najmniej 10, które są potrzebne do pokrycia zbioru podanego na wejściu.

Możesz założyć, że dane są dobrane tak, że ta liczba nie przekracza 5.

Ograniczenia

$1 \leq N \leq 500$, $1 \leq A_i \leq 1\,000\,000$.

Przykład

Wejście

```
12
7 12 18 22 29 42 51 62 100 102 200 500
```

Wyjście

```
3
```

Wyjaśnienie

W tym przykładzie możliwe jest pokrycie zbioru następującymi ciągami:

- $(12, 22, 32, \ldots, 102)$, pokrywamy nim elementy $12, 22, 42, 62, 102$,
- $(7, 18, 29, 40, 51)$, pokrywamy nim elementy $7, 18, 29, 51$,
- $(100, 200, 300, 400, 500)$, pokrywamy nim elementy $100, 200, 500$.