Gra karciana **(gra-karciana)**

Limit pamięci: 64 MB Limit czasu: 1.00 s

Jasio i Małgosia grają w grę karcianą. Część jej zasad brzmi następująco:

- Używana jest standardowa talia kart, która liczy 52 karty, po 4 z każdej figury.
- Figury, w kolejności od najsłabszej do najsilniejszej, to `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `T` (dziesiątka), `J` (walet), `Q` (dama), `K` (król), `A` (as).
- Karty o tych samych figurach mają równą siłę.
- Na początku gry każdy z graczy dobiera z potasowanej talii 3 karty, których nie pokazuje przeciwnikowi.

Znajomość reszty zasad nie jest potrzebna do rozwiązania tego zadania. Małgosia dobrała swoje trzy karty jako pierwsza i zastanawia się dla każdej z nich, ile różnych, silniejszych od niej kart może teraz dobrać Jasio. Napisz program, który obliczy to za nią.

Wejście

W pierwszym wierszu wejścia znajdują się trzy litery, oddzielone pojedynczymi znakami odstępu, oznaczające figury kart, które dobrała Małgosia.

Wyjście

W pierwszym wierszu wyjścia powinny znaleźć się trzy liczby, oddzielone pojedynczymi znakami odstępu. i-ta z nich powinna być równa liczbie kart, które Jasio może dobrać, i które są silniejsze od i-tej karty Małgosi.

Przykład

Wejście
```
2 3 4
```

Wyjście
```
46 43 40
```

Wyjaśnienie
W talii pozostały wszystkie karty silniejsze niż `4` – jest ich 40. Poza nimi są jeszcze trzy `4`, trzy `3`, i trzy `2`.

Wejście
```
A A A
```

Wyjście
```
0 0 0
```

Ciąg permutacji **(permutacje)**

Limit pamięci: 64 MB Limit czasu: 1.00 s

Jasio przygotowuje się do zawodów Bajtockiej Olimpiady Informatycznej Juniorów. Ostatnio nauczył się funkcji `next_permutation`, która generuje następną leksykograficznie permutację zadanego ciągu obiektów. W tym zadaniu będziemy rozpatrywali jedynie permutacje zbioru $\{1, 2, \ldots, N\}$, a więc dowolne ustawienie elementów (każdy element występuje dokładnie raz) tego zbioru w ciąg.

To bardzo fajnie, pomyślał Jasio – będzie można łatwo napisać rozwiązanie naiwne testujące wszystkie możliwości kolejności wykonania jakichś akcji. Jasiowi jednak nie do końca podoba się kolejność generowanych permutacji. Przykładowo: permutacja $(1, 5, 4, 3, 2)$ bezpośrednio poprzedza $(2, 1, 3, 4, 5)$. Jasio wolałby, aby sąsiednie dwie permutacje różniły się możliwie mało, najlepiej zamianą dokładnie dwóch sąsiednich elementów. Wtedy w jego naiwnych rozwiązaniach często możliwa jest tylko drobna aktualizacja wyniku poprzedniej permutacji, żeby uzyskać wynik dla następnej, zamiast obliczać ów wynik od nowa. Czy pomożesz mu dobrać lepszą kolejność, tak żeby wygerować wszystkie permutacje, każdą dokładnie jeden raz, ale żeby każde dwie sąsiednie permutacje różniły się jedynie zamianą pewnych dwóch sąsiednich elementów?

Napisz program, który wczyta liczbę naturalną N, wygeneruje wszystkie $N!$ różnych permutacji w odpowiedniej kolejności i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna N, określająca długość każdej permutacji.

Wyjście

Twój program powinien wypisać na wyjście dokładnie $N!$ permutacji zbioru $\{1, 2, \ldots, N\}$, po jednej w wierszu. Każda permutacja powinna być wypisana za pomocą N parami różnych liczb od 1 do N włącznie, poodzielanych pojedynczymi odstępami.

Wypisane permutacje mają być parami różne, a każde dwie sąsiednie wypisane permutacje mają się różnić pozycją dokładnie dwóch sąsiednich elementów.

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

Ograniczenia

$2 \leq N \leq 9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
3	1 2 3 1 3 2 3 1 2 3 2 1 2 3 1 2 1 3	Możliwe są różne poprawne wyjścia. Podajemy tutaj jedynie przykładowe poprawne rozwiązanie.

Programy telewizyjne **(programy-telewizyjne)**

Limit pamięci: 64 MB Limit czasu: 1.00 s

Jasio uwielbia oglądać telewizję, a szczególnie jego N ulubionych programów. Niestety, z racji tego, że nadawane są na różnych kanałach, czasami nachodzą na siebie w czasie. Jasio musi w takich sytuacjach podjąć trudną decyzję i zdecydować, który program obejrzy. Chłopak woli krótkie programy i stosuje zasadę, zgodnie z którą na pewno nie będzie oglądał programu P, takiego że istnieje inny program Q, który zaczyna się i kończy podczas trwania P. Innymi słowy, jeżeli istnieje para programów P, Q, taka że zachodzi *początek*$(P) \leq$ *początek*(Q) oraz *koniec*$(P) \geq$ *koniec*(Q), to Jasio nie obejrzy na pewno programu P.

Twoim zadaniem jest obliczenie liczby programów, które Jasio być może obejrzy.

Wejście

W pierwszym wierszu wejścia znajduje się liczba N, oznaczająca liczbę programów. i-ty z następnych N wierszy zawiera dwie liczby A_i, B_i, oznaczające czas początku i końca i-tego programu.

Przedziały czasowe wyznaczane przez programy są **parami różne**. To znaczy, że nie ma dwóch programów P, Q, dla których jednocześnie *początek*$(P) =$ *początek*(Q) oraz *koniec*$(P) =$ *koniec*(Q).

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się jedna liczba całkowita, oznaczająca liczbę programów, które Jasio być może obejrzy.

Ograniczenia

$1 \leq N \leq 1\,000\,000$, $|A_i|, |B_i| \leq 10^9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
5 1 5 2 6 3 7 4 6 4 8	2	Jasio nie obejrzy programów $(2, 6)$, $(3, 7)$, $(4, 8)$ ze względu na program $(4, 6)$, który się w nich czasowo zawiera.