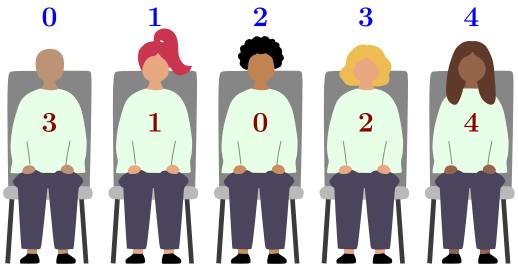
D. Plan miejsc (seatingplan)

W ceremonii zakończenia tego EGOI-a weźmie udział N ważnych gości. Wszyscy oni muszą zostać usadzeni w pierwszym rzędzie w ściśle określonej kolejności, która odpowiada wszystkim niuansom dyplomatycznego protokołu. Ustalenie prawidłowej kolejności miejsc zajęło Kalinie dwie bezsenne noce.

Daria nadzoruje ceremonię zakończenia. Jednym z jej wielu obowiązków jest upewnienie się, że miejsca w pierwszym rzędzie mają prawidłowe tabliczki z imionami. Jest tylko jeden malutki problem: Kalina nigdy nie przekazała jej prawidłowej kolejności miejsc, a teraz zaginęła. Na szczęście fotografka Patrycja ma aplikację, która może się przydać.

Patrycja musiała przygotować swoje aparaty tak, aby móc zrobić kilka konkretnych zdjęć gościom w pierwszym rzędzie. W ramach przygotowań musiała wiedzieć, jak szerokie będzie każde ze zdjęć, więc Kalina napisała dla niej aplikację, która szybko wypisuje potrzebne jej informacje. Daria chce teraz użyć tej aplikacji, aby odtworzyć poprawne przypisanie miejsc.

N ważnych gości jest ponumerowanych od 0 do $N-1$. Miejsca w pierwszym rzędzie również są ponumerowane od 0 do $N-1$, od lewej do prawej. Dla każdego I ($0 \le I \le N-1$), niech g_I oznacza gościa, który ma siedzieć na miejscu I, a s_I miejsce, w którym ma siedzieć gość I.



Rysunek 1: Rząd z pięcioma gośćmi. Dla tego rzędu $g = [3, 1, 0, 2, 4]$ i $s = [2, 1, 3, 0, 4]$.

Aplikacja działa następująco:

- Patrycja wprowadza numery I, J, K odpowiadające dokładnie trzem parami różnym gościom.
- Aplikacja odpowiada minimalną liczbą gości, którzy będą widoczni, jeśli wszyscy trzej wybrani goście znajdą się na zdjęciu. Formalnie, aplikacja wyświetli wartość $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Na przykład, rozważmy sytuację pokazaną na Rysunku 1:

- Goście $I = 0$, $J = 2$ i $K = 4$ siedzą na miejscach $s_I = 2$, $s_J = 3$ i $s_K = 4$. Jeśli Patrycja ich wybierze, aplikacja wyświetli wartość $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Innymi słowy, najwęższe zdjęcie zawierające gości 0, 2 i 4 obejmuje tylko tych trzech gości.

- Goście $I = 0$, $J = 4$ i $K = 3$ siedzą na miejscach $s_I = 2$, $s_J = 4$ i $s_K = 0$. Jeśli Patrycja ich wybierze, aplikacja wyświetli wartość $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Innymi słowy, zdjęcie zawierające wymienionych trzech gości musi obejmować wszystkich 5 gości.

Pomóż Darii wyznaczyć prawidłową kolejność miejsc, korzystając z aplikacji Patrycji. Mówiąc ściślej, Twój program powinien wyznaczyć i wypisać ciąg $g_0, g_1, ..., g_{N-1}$. Zawsze istnieją dokładnie dwie

poprawne odpowiedzi (jedna jest odwróceniem drugiej) i możesz wypisać dowolną z nich. Twój wynik będzie zależał od liczby zapytań do aplikacji, jakie wykonuje twoje rozwiązanie.

Implementacja

⇨ To jest zadanie interaktywne. Twój program będzie używał standardowego wejścia i wyjścia do komunikacji ze sprawdzaczką w formacie opisanym poniżej.

Twój program powinien rozpocząć od wczytania jednego wiersza wejścia zawierającego dodatnią liczbę całkowitą T, liczbę przypadków testowych, które po nim nastąpią.

Dla każdego przypadku testowego Twój program powinien rozpocząć od wczytania jednego wiersza wejścia zawierającego dodatnią liczbę całkowitą N, liczbę miejsc, która jest jednocześnie liczbą gości.

Aby zadać zapytanie, Twój program powinien wypisać wiersz w formacie „? I J K", gdzie $0 \leq I, J, K \leq N-1$ to trzy **parami różne** liczby.

Po zadaniu zapytania, Twój program powinien wczytać jeden wiersz zawierający jedną dodatnią liczbę całkowitą, będącą odpowiedzią na twoje zapytanie.

Aby udzielić odpowiedzi z poprawną kolejnością miejsc, Twój program powinien wypisać wiersz w formacie „! g_0 … g_{N-1}".

Po rozwiązaniu wszystkich T testów Twój program powinien zakończyć się w normalny sposób.

Zauważ, że oficjalna sprawdzaczka używana w systemie CMS do testowania Twojego rozwiązania może być **adaptacyjna**. Oznacza to, że dla niektórych testów permutacja gości nie jest z góry ustalona. Zamiast tego sprawdzaczka może zmieniać swoje zachowanie w zależności od konkretnych zapytań wysłanych przez Twój program.

Czyszczenie bufora (Flushing). Jeśli nie używasz dostarczonych szablonów, upewnij się, że czyścisz standardowe wyjście po wypisaniu każdej linii, w przeciwnym razie Twój program może zostać oceniony jako *Błędna odpowiedź*. W języku Python dzieje się to automatycznie, jeśli używasz `input()` do odczytu linii, ale możesz użyć `print(..., flush=True)` by wymusić flush. W C++, `cout << endl;` poza wypisaniem nowej linii czyści bufor; jeśli używasz printf, użyj `fflush(stdout)`.

Ograniczenia

- $1 \leq T \leq 10$.
- N wyniesie 5 (tylko w teście przykładowym), 8, 40 lub 2000.
- W każdym przypadku testowym możesz zadać co najwyżej 10 000 zapytań.

Punktacja

Zestaw testów dzieli się na następujące podzadania. Każde podzadanie składa się z jednej lub większej liczby testów. Żeby zdobyć punkty za podzadanie, musisz poprawnie rozwiązać wszystkie zawarte w nim testy.

- **Podzadanie 0 [0 punktów]**: Przykład ($N = 5$).
- **Podzadanie 1 [9 punktów]**: $N = 8$.
- **Podzadanie 2 [11 punktów]**: $N = 2000$, a goście 0 i 1 siedzą obok siebie.
- **Podzadanie 3 [15 punktów]**: $N = 40$.
- **Podzadanie 4 [65 punktów]**: $N = 2000$.

Za podzadania 1 i 2 każde rozwiązanie, które poprawnie rozwiąże wszystkie testy, otrzyma pełną pulę punktów.

Dla podzadań 3 i 4 Twoje rozwiązanie musi poprawnie rozwiązać wszystkie testy, aby zdobyć jakiekolwiek punkty, a Twój wynik będzie zależał od Q_s, największej liczby zapytań, których Twoje

rozwiązanie potrzebowało do rozwiązania pojedynczego testu. Niech $X_s = \max(1, Q_s/N)$. Punkty za podzadania 3 i 4 są wtedy obliczane następująco:

$$\text{wynik}_3 = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \qquad \text{wynik}_4 = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

Wartość wynik_s jest zaokrąglana do najbliższej liczby całkowitej dla każdego podzadania, a Twój końcowy wynik to ich suma. Aby uzyskać pełną pulę punktów, musisz rozwiązać podzadanie 3 w co najwyżej 55 zapytaniach, a podzadanie 4 w co najwyżej 2597 zapytaniach.

Przykładowe wartości Q_s i odpowiadające im wyniki w podzadaniach 3 i 4 są zilustrowane poniżej.

Q_s	55	56	60	70	80	100	150	10000
wynik_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
wynik_4	65	58	53	35	26	21	14	11

Przykłady

Sprawdzaczka	Rozwiązanie
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Wyjaśnienie

Przykładowe wejście zawiera jeden przypadek testowy ($T = 1$) z $N = 5$ gośćmi. Ukryta konfiguracja gości w tym teście odpowiada rysunkowi Rysunek 1.

Pierwsze zapytanie złożone przez przykładowe rozwiązanie to 0, 2, 4. Odpowiedź 3 na to zapytanie mówi nam, że ci goście siedzą, w jakiejś nieznanej kolejności, na trzech kolejnych miejscach obok siebie.

Odpowiedź 3 na drugie zapytanie mówi to samo dla gości 3, 0 i 1.

Możemy zatem wywnioskować, że gość 0 musi siedzieć na środku, z gośćmi 2 i 4 po jednej stronie, a gośćmi 1 i 3 po drugiej stronie.

Po trzecim zapytaniu możemy już być pewni, że goście muszą siedzieć albo w kolejności $[3, 1, 0, 2, 4]$, albo w kolejności odwróconej $[4, 2, 0, 1, 3]$. Możemy wypisać dowolną z nich.

Szablony rozwiązania i szczegóły oceny w CMS

Bardzo zachęcamy do używania przykładowego rozwiązania dla C++ i Pythona. Sprawdza ono, czy komunikacja ze sprawdzaczką była prawidłowa, i kończy się jeżeli nie była.

Sprawdzaczka komunikująca się z Twoim rozwiązaniem zwraca pierwszy błąd, jaki napotka i się kończy. Jeżeli nie wykorzystasz danego przykładowego rozwiązania, Twoje rozwiązanie może wtedy zakończyć się błędem lub czekać na odpowiedź. Jeżeli tak się stanie, w większości przypadków otrzymasz niepomocną odpowiedź na CMS, na przykład „Execution timed out (wall clock limit exceeded)".

Polecamy również używania narzędzia do testowania (zobacz poniżej) do testowania Twojego rozwiązania lokalnie przed submitowaniem go. Narzędzie do testowania sprawdza wyjście Twojego programu i raportuje naruszenia protokołu.

Narzędzie do testowania

Aby ułatwić ci testowanie rozwiązania, udostępniamy prosty program, który możesz pobrać z systemu CMS. Korzystanie z narzędzia jest opcjonalne. Zauważ, że oficjalna sprawdzaczka w systemie CMS różni się od tego narzędzia do testowania.

Aby użyć narzędzia, potrzebujesz pliku wejściowego. Możesz użyć załączonego pliku przykładowego `seatingplan.input0.txt` lub stworzyć własny. Plik wejściowy powinien zaczynać się od linii z liczbą testów T, a następnie powinien zawierać dwie linie na test: jedną linię z liczbą N a potem jedną linię z liczbami $g_0, g_1, ..., g_{N-1}$.

Dla programów w Pythonie, na przykład `seatingplan.py` (zazwyczaj uruchamianych przez `pypy3 seatingplan.py`) uruchom narzędzie testujące w następujący sposób:

```
python3 testing_tool.py pypy3 seatingplan.py < seatingplan.input0.txt
```

Dla programów w C++, najpierw skompiluj swoje rozwiązanie:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o seatingplan seatingplan.cpp
```

a następnie uruchom narzędzie do testowania:

```
python3 testing_tool.py ./seatingplan < seatingplan.input0.txt
```