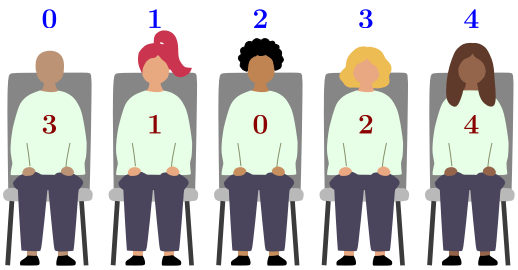
D. Platstilldelning (seatingplan)

Avslutningsceremonin för årets EGOI kommer att gästas av N viktiga personer. Alla dessa måste sitta på första raden i en väldigt specifik ordning som tar hänsyn till alla diplomatiska nyanser. Att lista ut den rätta ordningen tog Ellinor två sömnlösa nätter.

Ruth är ansvarig för avslutningsceremonin. Ett av hennes många ansvarsområden är att se till att platserna på första raden har rätt namnskyltar. Det finns bara ett litet problem: Ellinor berättade aldrig den korrekta ordningen till Ruth, och nu går Ellinor inte att hitta någonstans. Som tur är har fotografen Lovisa en app som kanske kan vara till hjälp.

Lovisa behövde förbereda sina kameror så att hon kunde ta några specifika bilder av gästerna på första raden. För att ställa in kamerorna behövde hon veta hur bred varje bild skulle bli, så Lovisa gjorde en app åt henne som snabbt ger den informationen. Ruth vill nu använda appen för att hitta rätt bordsplacering.

De N viktiga gästerna är numrerade från 0 till $N-1$. Platserna på första raden är också numrerade från 0 till $N-1$, från vänster till höger. För varje I ($0 \leq I \leq N-1$), låt g_I vara gästen som ska sitta på plats I, och låt s_I vara platsen där gäst I ska sitta.



Figur 1: En rad med fem gäster. För denna rad är $g = [3, 1, 0, 2, 4]$ och $s = [2, 1, 3, 0, 4]$.

Appen funkar så här:

- Lovisa knappar in numren I, J, K av exakt tre olika gäster.
- Appen berättar för henne det minsta antalet gäster som kommer synas om alla tre valda gäster är med på bilden. Formellt visar appen värdet $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Till exempel, betrakta situationen i Figur 1:

- Gästerna $I = 0$, $J = 2$ och $K = 4$ sitter på platserna $s_I = 2$, $s_J = 3$ och $s_K = 4$. Om Lovisa väljer dem visar appen värdet $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Med andra ord, det smalaste fotot som innehåller gästerna 0, 2 och 4 innehåller bara de tre gästerna.

- Gästerna $I = 0$, $J = 4$ och $K = 3$ sitter på platserna $s_I = 2$, $s_J = 4$ och $s_K = 0$. Om Lovisa väljer dem visar appen värdet $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Med andra ord, ett foto som innehåller de tre gästerna måste innehålla alla 5 gäster.

Hjälp Ruth att lista ut rätt bordsplacering med hjälp av Lovisas app. Mer specifikt ska ditt program hitta och skriva ut sekvensen $g_0, g_1, ..., g_{N-1}$. Det finns alltid exakt två rätta svar (det ena är det andra svaret baklänges), och du kan skriva ut vilken som helst av dem. Din poäng beror på hur många frågor din lösning ställer till appen.

Implementering

⇨ Det här är ett interaktivt problem. Ditt program kommer använda standard in och ut för att prata med en domare i formatet som beskrivs nedan.

Ditt program ska börja med att läsa en rad från indatan som innehåller ett positivt heltal T, antalet testfall som följer.

För varje testfall ska ditt program börja med att läsa en rad som innehåller ett positivt heltal N, antalet platser, vilket också är antalet gäster.

För att ställa en fråga ska ditt program skriva ut en rad på formatet "? I J K", där $0 \leq I, J, K \leq N-1$ är tre **olika** tal.

Efter att ha ställt en fråga ska ditt program läsa in en rad som innehåller ett positivt heltal, svaret på din fråga.

För att svara med en korrekt bordsplacering ska ditt program skriva ut en rad på följande format: "! g_0 … g_{N-1}".

När alla T testfall är lösta ska ditt program avslutas normalt.

Notera att den officiella domaren i CMS som testar din lösning kan vara **adaptiv**. Det betyder att i vissa testfall är gästernas ordning inte bestämd på förhand. Istället kan domaren vilken av de kvarvarande permutationerna den använder beroende på de frågor ditt redan ställt.

Flushning. Om du inte använder de tillhandahållna mallarna, se till att flusha standardutmatningen efter att du skrivit ut varje rad, annars kan ditt program bedömas som *Not correct*. I Python händer detta automatiskt om du använder `input()` för att läsa rader, och du kan använda `print(..., flush=True)` för att tvinga den den att flusha. I C++ flushar `cout << endl;` och skriver ut en ny rad samtidigt; om du använder printf, använd `fflush(stdout)`.

Begränsningar

- $1 \leq T \leq 10$.
- N kommer vara 5 (endast exempelfallet), 8, 40 eller 2000.
- För varje testfall får du ställa högst 10 000 frågor.

Poängsättning

Ditt program kommer att testas på flera testfall som är uppdelade i testgrupper. För att få poäng för en testgrupp måste du klara alla testfall i den gruppen.

- **Testgrupp 0 [0 poäng]**: Exempel ($N = 5$).
- **Testgrupp 1 [9 poäng]**: $N = 8$.
- **Testgrupp 2 [11 poäng]**: $N = 2000$, och gästerna 0 och 1 sitter bredvid varandra.
- **Testgrupp 3 [15 poäng]**: $N = 40$.
- **Testgrupp 4 [65 poäng]**: $N = 2000$.

För testgrupp 1 och 2 får du full poäng om du löser alla testfall korrekt, så länge de inte använder mer än 10000 frågor.

För testgrupp 3 och 4 måste din lösning klara alla testfall för att få poäng, och din poäng beror på Q_s, det största antalet frågor din lösning behövde ställa för att lösa ett testfall i den testgruppen. Låt $X_s = \max(1, Q_s/N)$. Poängen för testgrupp 3 och 4 räknas sen ut så här (se nästa sida):

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

Värdet på score_s avrundas till närmaste heltal för varje testgrupp, och din totala poäng är summan av dessa. För att få full poäng måste du klara testgrupp 3 med som mest 55 frågor och testgrupp 4 med som mest 2597 frågor. Exempelvärden av Q_s och motsvarande poäng för testgrupper 3 och 4 visas nedan.

Q_s	55	56	60	70	80	100	150	10000
score_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
score_4	65	58	53	35	26	21	14	11

Exempel

Domare	Lösning
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Förklaring

Exempelindatan innehåller ett testfall ($T = 1$) med $N = 5$ gäster. Den hemliga ordningen av gäster i det här testfallet motsvarar Figur 1.

Den första frågan som exempellösningen ställer är 0, 2, 4. Svaret 3 på den frågan säger oss att dessa gäster sitter, i någon okänd ordning, på tre platser i rad bredvid varandra.

Svaret 3 på den andra frågan säger oss samma sak om gästerna 3, 0 och 1.

Vi kan nu lista ut att gäst 0 måste sitta i mitten, med gästerna 2 och 4 på ena sidan och gästerna 1 och 3 på andra sidan.

Efter den tredje frågan kan vi redan vara säkra på att gästerna måste sitta antingen i ordningen $[3, 1, 0, 2, 4]$ eller baklänges $[4, 2, 0, 1, 3]$. Vi kan skriva ut vilken som helst av dem.

Kodmallar och utvärderingsdetaljer i CMS

Vi rekommenderar starkt att använda de tillhandahållna kodmallarna för C++ och Python. Dessa kontrollerar om kommunikationen med bedömaren lyckades och avslutar programmet på ett kontrollerat sätt om den inte gjorde det.

Om du inte använder de tillhandahållna mallarna kan CMS, i fall där din lösning är felaktig, visa fel domslut. Till exempel kan du få ”Execution killed by signal” eller ”Execution timed out (wall clock limit exceeded)” i stället för ”Output isn’t correct”.

Vi rekommenderar också testverktyget (se nedan) för att testa din lösning lokalt innan du skickar in den. Testverktyget kontrollerar utdata från din lösning och rapporterar användning av ogiltiga frågor.

Testverktyg

För att göra det lättare att testa din lösning har vi lagt upp ett enkelt verktyg som du kan ladda ner från CMS. Verktyget är helt frivilligt att använda. Notera att den riktiga domaren i CMS är annorlunda från testverktyget.

För att använda verktyget behöver du en indatafil. Du kan använda exempelfilen `seatingplan.input0.txt` eller göra en egen. Indatafilen ska börja med en rad med antalet testfall T, och sedan ska den ha två rader per testfall: en rad med antalet N och sedan en rad med talen $g_0, g_1, \cdots, g_{N-1}$.

För Python-program, exempelvis `seatingplan.py` (körs normalt som `pypy3 seatingplan.py`), kör testverktyget så här:

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

För C++-program, kompilera först din lösning:

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

och kör sedan testverktyget:

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