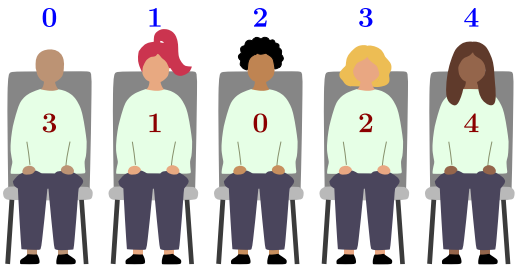
D. ადგილების განაგება (seatingplan)

EGOI-ს დახურვის ცერემონიას N რაოდენობა მნიშვნელოვანი სტუმარი დაესწრება. ყველა მათგანი წინა რიგში უნდა დაჯდეს ძაღიან კონკრეტული წესით, რომელიც დიპლომატიური პროტოკოლის ყველა ნიუანსს შეესაბამება. სწორი განაგების დადგენას ნოემიმ ორი უძიღო ღამე შეაღია.

ვერონიკა დახურვის ცერემონიაღს ხელმძღვანელობს. მისი ერთ-ერთი მოვაღეობაა, დარწმუნდეს, რომ წინა რიგის სკამებზე სახელების სწორი წარწერებია. მხოღოდ ერთი პატარა პრობღემაა: ნოემიმ არ უთხრა მას სწორი განაგება და ახღა ნოემი ვერსად იპოვა. საბედნიეროდ, ფოტოგრაფ დორკას აქვს აპლიკაცია, რომელიც შეიძლება გამოადგეს.

დორკას კამერების მომზადება დასჭირდა, რათა პირველ რიგში მსხდომი სტუმრების კონკრეტული ფოტოები გადაეღო. მომზადებისთვის მას აინტერესებდა, თუ რა სიგანის იქნებოდა თითოეული ფოტო, ამიტომ ნოემიმ შეუქმნა აპლიკაცია, რომელიც სწრაფად გასცემს მისთვის საჭირო ინფორმაციას. ახლა ვერონიკას უნდა ამ აპლიკაციის გამოყენება, რათა იპოვოს ადგილების სწორი განაგება.

N ცაღი მნიშვნელოვანი სტუმარი დანომრილია 0-დან $(N-1)$-მდე. წინა რიგის სკამებიც დანომრილია 0-დან $(N-1)$-მდე მარცხნიდან მარჯვნივ. თითოეული I-სთვის $(0 \leq I \leq N-1)$, დავუშვათ, g_I არის სტუმარი, რომელიც I ადგილზე უნდა იჯდეს, ხოღო s_I არის ადგილი, სადაც I სტუმარი უნდა იჯდეს.



სურ. 1: სტუმრების რიგი ხუთი ადამიანით. ამ რიგისთვის, $g = [3, 1, 0, 2, 4]$ და $s = [2, 1, 3, 0, 4]$.

აპლიკაცია მუშაობს შემდეგნაირად:

- დორკას შეაქვს სამი განსხვავებული სტუმრის ნომრები: I, J, K.
- აპლიკაცია ეუბნება სტუმრების მინიმაღურ რაოდენობას, რომლებიც გამოჩნდება, თუ სამივე არჩეული სტუმარი იქნება ფოტოში. ფორმაღურად, აპლიკაცია აჩვენებს მნიშვნელობას $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

მაგაღითად, შეხედე სურ. 1 -ში ნაჩვენებ სიტუაციას:

- სტუმრები $I = 0$, $J = 2$ და $K = 4$ სხედან $s_I = 2$, $s_J = 3$ და $s_K = 4$ ადგილებზე. თუ დორკა მათ აირჩევს, აპლიკაცია გამოიტანს მნიშვნელობას $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  სხვა სიტყვებით რომ ვთქვათ, ყველაზე ვიწრო ფოტო, რომელიც შეიცავს სტუმრებს $0, 2$ და 4, ზუსტად ამ სამ სტუმარს მოიცავს.

- სტუმრები $I = 0$, $J = 4$ და $K = 3$ სხედან $s_I = 2$, $s_J = 4$ და $s_K = 0$ ადგილებზე. თუ დორკა მათ აირჩევს, აპლიკაცია გამოიტანს მნიშვნელობას $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

სხვა სიტყვებით რომ ვთქვათ, ფოტო, რომელიც შეიცავს მოცემულ სამ სტუმარს, აუცილებლად უნდა შეიცავდეს ყველა 5 სტუმარს.

დაეხმარე ვერონიკას ადგილების სწორი განლაგების დადგენაში დორკას აპლიკაციის გამოყენებით. უფრო კონკრეტულად, შენმა პროგრამამ უნდა დაადგინოს და გამოიტანოს მიმდევრობა $g_0, g_1, ..., g_{N-1}$. ყოველთვის არსებობს ზუსტად ორი სწორი პასუხი (ერთი მეორის შებრუნებულია) და შეგიძლია ნებისმიერი მათგანი გამოიტანო. შენი ქულა დამოკიდებული იქნება იმაზე, თუ რამდენ მოთხოვნას გაუგზავნი აპლიკაციას შენი ამოხსნის ფარგლებში.

იმპლემენტაცია

> ⇨ ეს არის ინტერაქტიული ამოცანა. შენი პროგრამა გამოიყენებს სტანდარტულ შეტანას და გამოტანას შემაფასებელთან კომუნიკაციისთვის ქვემოთ აღწერილი ფორმატით.

შენმა პროგრამამ უნდა დაიწყოს შეტანიდან ერთი ხაზის წაკითხვით, რომელიც შეიცავს მთელ დადებით T რიცხვს — სატესტო შემთხვევების რაოდენობას.

თითოეული სატესტო შემთხვევისთვის შენმა პროგრამამ უნდა დაიწყოს შეტანიდან ერთი ხაზის წაკითხვით, რომელიც შეიცავს მთელ დადებით N რიცხვს — სკამების რაოდენობას, რაც ასევე სტუმრების რაოდენობასაა.

მოთხოვნის გასაგზავნად, შენმა პროგრამამ უნდა გამოიტანოს სტრიქონი ფორმატით: "? I J K", სადაც I, J, K $(0 \leq I, J, K \leq N-1)$ სამი **განსხვავებული** რიცხვია.

მოთხოვნის გაგზავნის შემდეგ, შენმა პროგრამამ უნდა წაიკითხოს ერთი ხაზი, რომელიც შეიცავს ერთ მთელ დადებით რიცხვს — პასუხს შენს მოთხოვნაზე.

სწორი განლაგების პასუხად, შენმა პროგრამამ უნდა გამოიტანოს სტრიქონი ფორმატით: "! g_0 ⋯ g_{N-1}".

ყველა T სატესტო შემთხვევის ამოხსნის შემდეგ შენმა პროგრამამ ნორმალურად უნდა დაასრულოს მუშაობა.

გაითვალისწინე, რომ CMS-ში გამოყენებული ოფიციალური შემფასებელი შეიძლება იყოს **ადაპტირებადი**. ეს ნიშნავს, რომ ზოგიერთი სატესტო შემთხვევისთვის სტუმრების პერმუტაცია წინასწარ არ არის განსაზღვრული. სანაცვლოდ, შემფასებელმა შესაძლოა გადაწყვიტოს, რომელი დარჩენილი პერმუტაცია გამოიყენოს იმის მიხედვით, თუ რა მოთხოვნებს გააგზავნის შენი პროგრამა.

ბუფერის გასუფთავება (Flushing). თუ არ იყენებ მოწოდებულ შაბლონებს, დარწმუნდი, რომ სტანდარტული გამოტანის ბუფერს ასუფთავებ ყოველი ხაზის გამოტანის შემდეგ, წინააღმდეგ შემთხვევაში შენი პროგრამა შეიძლება შეფასდეს როგორც *არასწორი*. Python-ში ეს ავტომატურად ხდება, თუ სტრიქონების წასაკითხად იყენებ `input()`-ს, ხოლო `print(..., flush=True)`-ის გამოყენებით შეგიძლია იძულებით გაასუფთავო ბუფერი. C++-ში `cout << endl;` ბეჭდვასთან ერთად აკეთებს ბუფერის გასუფთავებასაც; თუ იყენებ `printf`-ს, გამოიყენე `fflush(stdout)`.

შეზღუდვები

- $1 \leq T \leq 10$.
- N იქნება 5 (მხოლოდ მაგალითისთვის), 8, 40 ან 2000.
- თითოეული სატესტო შემთხვევისთვის შეგიძლია გააკეთო მაქსიმუმ 10 000 მოთხოვნა.

შეფასება

შენი პროგრამა შემოწმდება რამდენიმე სატესტო შემთხვევაზე, რომლებიც გაერთიანებულია ქვეამოცანებში. ქვეამოცანისთვის ქულის მისაღებად, თქვენი ამოხსნა სწორ პასუხს უნდა იძლეოდეს ამ ჯგუფში შემავალ თითოეულ ტესტზე.

- **ქვეამოცანა 0** [**0 ქულა**]: მაგალითი ($N = 5$).
- **ქვეამოცანა 1** [**9 ქულა**]: $N = 8$.
- **ქვეამოცანა 2** [**11 ქულა**]: $N = 2000$, და სტუმრები 0 და 1 სხედან ერთმანეთის გვერდით.
- **ქვეამოცანა 3** [**15 ქულა**]: $N = 40$.
- **ქვეამოცანა 4** [**65 ქულა**]: $N = 2000$.

ქვეამოცანებისთვის 1 და 2, ნებისმიერი ამოხსნა, რომელიც სწორ პასუხს იძლევა ყველა ტესტზე, მიიღებს სრულ ქულას.

ქვეამოცანებისთვის 3 და 4 შენმა ამოხსნამ სწორად უნდა ამოხსნას ყველა სატესტო შემთხვევა, რათა მიიღოს რაიმე ქულა, ხოლო შენი ქულა დამოკიდებული იქნება Q_s-ზე — მოთხოვნების მაქსიმალურ რაოდენობაზე, რომელიც შენს ამოხსნას დასჭირდა სატესტო შემთხვევის ამოსახსნელად. ვთქვათ $X_s = \max\left(1, \frac{Q_s}{N}\right)$. ქვეამოცანებისთვის 3 და 4 ქულები გამოითვლება შემდეგნაირად:

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

score_s მნიშვნელობა მრგვალდება უახლოეს მთელ რიცხვამდე თითოეული ქვეამოცანისთვის, ხოლო შენი საერთო ქულა არის მათი ჯამი. სრული ქულის მისაღებად ქვეამოცანა 3 უნდა ამოხსნა მაქსიმუმ 55 მოთხოვნით, ხოლო ქვეამოცანა 4 მაქსიმუმ 2597 მოთხოვნით. Q_s-ის და ქვეამოცანებისთვის 3 და 4 ქულების მაგალითები ნაჩვენებია ქვემოთ.

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

მაგალითები

Grader	ამოხსნა
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

განმარტება

მაგაღითის შეტანა შეიცავს ერთ სატესტო შემთხვევას ($T = 1$) $N = 5$ სტუმრით. ამ სატესტო შემთხვევაში სტუმრების დაფარუღი კონფიგურაცია შეესაბამება სურ. 1 -ს.

მაგაღითი ამოხსნის მიერ გაგზავნიღი პირვეღი მოთხოვნაა $0, 2, 4$. ამ მოთხოვნაზე პასუხი 3 გვეუბნება, რომ ეს სტუმრები სხედან უცნობი თანმიმდევრობით, სამ მიმდებარე სკამზე, ერთმანეთის გვერდიგვერდ.

მეორე მოთხოვნაზე პასუხი 3 იგივეს გვეუბნება სტუმრებზე 3, 0 და 1.

ახღა შეგვიძღია დავასკვნათ, რომ სტუმარი 0 შუაში უნდა იჯდეს, სტუმრები 2 და 4 ერთ მხარეს, ხოღო სტუმრები 1 და 3 მეორე მხარეს.

მესამე მოთხოვნის შემდეგ, უკვე შეგვიძღია დარწმუნებუღნი ვიყოთ, რომ სტუმრები უნდა ისხდნენ ან $[3, 1, 0, 2, 4]$ მიმდევრობით, ან შებრუნებუღი $[4, 2, 0, 1, 3]$ მიმდევრობით. ჩვენ შეგვიძღია გამოვიტანოთ ნებისმიერი ამ მიმდევრობიდან.

კოდის შაბღონები და CMS-ში შეფასების დეტაღები

მკაცრად გირჩევთ გამოიყენოთ C++-ისა და Python-ისთვის მოწოდებუღი კოდის შაბღონები. ისინი ამოწმებენ, იყო თუ არა კომუნიკაცია შემაფასებეღთან წარმატებუღი და კორექტუღად წყვეტენ მუშაობას, თუ არ იყო.

თუ არ იყენებ მოწოდებუღ შაბღონებს, იმ შემთხვევებში, როდესაც შენი ამოხსნა არასწორია, CMS-მა შესაძღოა აჩვენოს არასწორი ვერდიქტი. მაგაღითად, "Output isn't correct"-ის ნაცვღად შეიძღება მიიღო "Execution killed by signal" ან "Execution timed out (wall clock limit exceeded)".

ასევე გირჩევთ გამოიყენოთ სატესტო ინსტრუმენტი (იხ. ქვემოთ), რათა შეამოწმო შენი ამოხსნა ღოკაღურად ჩაბარებამდე. სატესტო ინსტრუმენტი ამოწმებს შენი ამოხსნის მიერ გამოტანიღ შედეგებს და გაცნობებს არასწორი მოთხოვნების გამოყენების შესახებ.

სატესტო ინსტრუმენტი

შენი ამოხსნის ტესტირების გასამარტივებღად გთავაზობთ მარტივ ინსტრუმენტს, რომღის ჩამოტვირთვაც შეგიძღია CMS-დან. ინსტრუმენტის გამოყენება არჩევითია. გაითვაღისწინე, რომ CMS-ზე არსებუღი ოფიციაღური შემფასებეღი განსხვავდება ამ სატესტო ინსტრუმენტისგან.

ინსტრუმენტის გამოსაყენებღად დაგჭირდება შეტანის ფაიღი. შეგიძღია გამოიყენო მოწოდებუღი მაგაღითის შეტანა `seatingplan.input0.txt` ან შექმნა შენი საკუთარი. შეტანის ფაიღი უნდა იწყებოდეს ხაზით, რომეღიც შეიცავს სატესტო შემთხვევების T რაოდენობას, შემდეგ კი თითოეუღ სატესტო შემთხვევაზე უნდა იყოს ორი ხაზი: ერთი ხაზი N-ით, და მეორე ხაზი სტუმრების თანმიმდევრობით $g_0, g_1, ..., g_{N-1}$.

Python-ის პროგრამებისთვის, ვთქვათ `seatingplan.py` (რომეღიც ჩვეუღებრივ ეშვება `pypy3 seatingplan.py`-ით), გაუშვი სატესტო ინსტრუმენტი შემდეგნაირად:

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

C++ პროგრამებისთვის, ჯერ დააკომპიღირე შენი ამოხსნა:

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

შემდეგ კი გაუშვი სატესტო ინსტრუმენტი:

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