სასწორი

ამინას აქვს ექვსი მონეტა, რომლებიც გადანომრილია **1**–დან **6**–მდე. მან იცის, რომ ყველა მონეტას განსხვავებული წონა აქვს და სურს, რომ დაალაგოს ისინი წონების მიხედვით. საამისოდ ამინა იყენებს ახალი ტიპის თეფშებიან სასწორს.

ტრადიციულ თეფშებიან სასწორს ორი თეფში აქვს. მის გამოსაყენებლად თითოეულ თეფშზე ათავსებენ თითო მონეტას და სასწორი განსაზღვრავს, თუ რომელი მათგანია უფრო მძიმე.

ამინას ახალი სასწორი უფრო რთულია. მას ოთხი თეფში აქვს, რომლებიც აღნიშნულია როგორც A, B, C და D. სასწორს აქვს ოთხი განსხვავებული ფუნქცია, რომელთაგან თითოეული პასუხობს განსხვავებულ შეკითხვებს მონეტების შესახებ. სასწორის გამოსაყენებლად საჭიროა ამინამ თითო მონეტა მოათავსოს A, B და C თეფშებზე. გარდა ამისა, მას შეუძლია მეოთხე – D თეფშზეც მოათავსოს ზუსტად ერთი მონეტა.

ამ ოთხი პარამეტრით სასწორის საშუალებით შეგვიძლია მივიღოთ პასუხი შემდეგ ოთხ შეკითხვაზე:

1. A, B და C თეფშებიდან რომელზე დევს ყველაზე მძიმე მონეტა?
2. A, B და C თეფშებიდან რომელზე დევს ყველაზე მსუბუქი მონეტა?
3. A, B და C თეფშებიდან რომელზე დევს მონეტა მედიანის მნიშვნელობით? (ანუ რომელი მონეტა არ წარმოადგენს არც ყველაზე მძიმეს და არც ყველაზე მსუბუქს)
4. A, B და C თეფშებიდან განიხილება მხოლოდ ის მონეტები, რომლებიც უფრო მძიმეა ვიდრე D თეფშზე მოთავსებული მონეტა. თუკი ასეთი მონეტები არსებობს, რომელია მათგან ყველაზე მსუბუქი? თუკი ასეთი მონეტები არ არსებობს, რომელია A, B და C თეფშებზე მოთავსებული მონეტებიდან ყველაზე მსუბუქი?

ამოცანა

დაწერეთ პროგრამა, რომელიც დაალაგებს ამინას 6 მონეტის წონებს ზრდადობით. პროგრამას შეუძლია შეკითხვით მიმართოს ამინას სასწორს, რათა მან შეადაროს მონეტათა წონები. თქვენს პროგრამას მიეწოდება რამდენიმე ტესტი, რომელთაგან თითოეული შეესაბამება ექვსი მონეტისაგან შედგენილ ახალ სიმრავლეს.

თქვენმა პროგრამამ უნდა მოახდინოს `init` და `orderCoins` ფუნქციების იმპლემენტაცია. თქვენი პროგრამის ყოველი გაშვებისას გრადერმა პირველ რიგში უნდა გამოიძახოს ფუნქცია `init` ზუსტად ერთხელ. ეს მოგაწოდებთ თქვენ ტესტების რაოდენობას, საშუალებას მოგცემთ ნებისმიერი რაოდენობის ცვლადთა ინიციალიზაცია მოახდინოთ. ყოველი ტესტისთვის გრადერი

გამოიძახებს ფუნქცია `orderCoins()`–ს მხოლოდ ერთხელ.

- `init(T)`
  - `T`: წარმოადგენს ტესტების რაოდენობას, რომლიც უნდა ამოხსნას თქვენმა პროგრამამ ერთი გაშვებისას. `T` არის მთელი რიცხვი დიაპაზონიდან $1, \ldots, 18$.
  - ამ ფუნქციამ მნიშვნელობა არ უნდა დააბრუნოს.
- `orderCoins()`
  - ეს ფუნქცია გამოძახებულ უნდა იქნას ყოველი ტესტისათვის ერთხელ.
  - ფუნქციამ უნდა განსაზღვროს ამინას მონეტების კორექტული დალაგება გრადერის მიერ `getLightest()`, `getHeaviest()`, `getMedian()` და/ან `getNextLightest()` ფუნქციების გამოძახებით.
  - როცა ფუნქცია დაადგენს კორექტულ დალაგებას, მან უნდა გამოიძახოს ფუნქცია `answer()`.
  - `answer()` ფუნქციის გამოძახების შემდეგ `orderCoins()` ფუნქციამ მუშაობა უნდა დაასრულოს. მას არ აქვს დასაბრუნებელი მნიშვნელობა.

თქვენს პროგრამაში შეგიძლიათ გამოიყენოთ შემდეგი ფუნქციები.

- `answer(C)` — თქვენმა პროგრამამ უნდა გამოიყენოს ეს ფუნქცია პასუხის დასაბრუნებლად.
  - `C`: 6 სიგრძის მქონე მასივი, რომელიც შეიცავს მონეტების კორექტულ დალაგებას. `C[0]`–დან `C[5]`–მდე ელემენტები უნდა შეიცავდნენ მონეტების დალაგებულ ნომრებს (ანუ, რიცხვებს 1–დან 6–მდე). დალაგება უნდა მოხდეს მსუბუქიდან მძიმისაკენ.
  - თქვენმა პროგრამამ ეს ფუნქცია უნდა გამოიძახოს ფუნქციიდან `orderCoins()`, თითოჯერ თითო ტესტისათვის.
  - ეს ფუნქცია მნიშვნელობას არ აბრუნებს.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — ფუნქციები შეესაბამებიან ამინას სასწორის 1, 2 და 3 ოფციებს.
  - `A`, `B`, `C`: წარმოადგენენ მონეტებს, რომლებიც დევს A, B და C თეფშებზე შესაბამისად. `A`, `B`, და `C` არის სამი განსხვავებული მთელი რიცხვი, რომელთაგან თითოეული მოთავსებულია 1–დან 6–მდე ჩათვლით.
  - ყოველი ფუნქცია აბრუნებს `A`, `B` და `C` რიცხვებიდან ერთ–ერთს: შესაბამისი მონეტის ნომერს. მაგალითად, `getHeaviest(A, B, C)` აბრუნებს მოცემული სამი მონეტიდან უმძიმესის ნომერს.
- `getNextLightest(A, B, C, D)` — ფუნქცია შეესაბამება ამინას სასწორის მე–4 ოფციას.
  - `A`, `B`, `C`, `D` წარმოადგენენ მონეტებს, რომლებიც დევს A, B, C და D

თეფშებზე შესაბამისად. `A`, `B`, `C` და `D` არის ოთხი განსხვავებული მთელი რიცხვი, რომელთაგან თითოეული მოთავსებულია **1**–დან **6**–მდე ჩათვლით.

- ფუნქცია აბრუნებს `A`, `B` და `C` რიცხვებიდან ერთ–ერთს: იმ მონეტის ნომერს, რომელიც შერჩეულია ზემოთ აღწერილი წესის შესაბამისად მეოთხე ოფციისათვის. ანუ, აბრუნებს A, B და C თეფშებიდან იმ მონეტის მნიშვნელობას, რომელიც ყველაზე მსუბუქია D მონეტაზე მძიმე მონეტებს შორის და თუკი ასეთი მონეტები არ არსებობს, მაშინ აბრუნებს A, B და C თეფშებზე მოთავსებული მონეტებიდან ყველაზე მსუბუქის მნიშვნელობას.

შეფასება

ამ ამოცანებს არ გააჩნიათ ქვეამოცანები. თქვენი შეფასება დამოკიდებული იქნება აწონვათა რაოდენობაზე (ანუ გრადერის მიერ `getLightest()`, `getHeaviest()`, `getMedian()` და/ან `getNextLightest()` ფუნქციათა ჯამურ გამოძახებაზე), რომელსაც თქვენი პროგრამა გააკეთებს.

თქვენი პროგრამა შესრულდება რამდენჯერმე რამდენიმე ტესტისათვის ყოველ გაშვებაზე. აღვნიშნოთ r–ით თქვენი პროგრამის გაშვებათა რაოდენობა. ეს რიცხვი ფიქსირებულია სატესტო მონაცემების მიხედვით. თუ თქვენი პროგრამა ვერ დაალაგებს მონეტებს ყველა ტესტისათვის ერთ გაშვებაზე, მიიღებთ 0 ქულას. წინააღმდეგ შემთხვევაში შეფასება იქნება ინდივიდუალური შემდეგი წესით:

ვთქვათ Q არის აწონვათა მინიმალური რაოდენობა, რომლითაც შესაძლებელია 6 ნებისმიერი მიმდევრობით მოცემული მონეტის დალაგება ამინას სასწორით. ამოცანის სირთულის შესანარჩუნებლად აქ Q–ს მნიშვნელობას არ გავხსნით.

ვთქვათ, თქვენი პროგრამის მიერ ყველა გაშვების ყველა ტესტს შორის აწონვათა უდიდესი რაოდენობა ტოლია $Q + y$ რაიმე მთელი y–სათვის. ახლა განვიხილოთ თქვენი პროგრამის რომელიმე გაშვება. ვთქვათ, მიმდინარე გაშვებაში ყველა ტესტში აწონვების მაქსიმალური რაოდენობაა $Q + x$ რაიმე არაუარყოფითი მთელი x რიცხვისათვის (თუ თქვენს მიერ შესრულებული აწონვების რაოდენობა ყველა ტესტში Q–ზე ნაკლებია, მაშინ $x = 0$.) შედეგად, ქულა, რომელსაც მიიღებთ ამ გაშვებისას, იქნება $\frac{100}{r((x+y)/5+1)}$. შედეგი დამრგვალდება ნაკლებობით, ორი ციფრის სიზუსტით მძიმის შემდეგ.

კერძოდ, თუ თქვენი პროგრამა მოახდენს არაუმეტეს Q აწონვას ყველა გაშვების ყველა ტესტში, თქვენ მიიღებთ 100 ქულას.

მაგალითი

ვთქვათ მონეტები დალაგებულია წონის მიხედვით **3 4 6 2 1 5** მსუბუქიდან მძიმისაკენ.

ფუნქცია	შედეგები	განმარტება
getMedian(4, 5, 6)	6	მონეტა **6** არის მედიანა **4**, **5** და **6** მონეტებში.

ფუნქცია	შედეგები	განმარტება
getHeaviest(3, 1, 2)	1	მონეტა 1 არის უმძიმესი 1, 2 და 3 მონეტებში.
getNextLightest(2, 3, 4, 5)	3	მონეტები 2, 3 და 4 მონეტები არიან მსუბუქები 5 მონეტაზე, მათ შორის ყველაზე მსუბუქია (3) მონეტა.
getNextLightest(1, 6, 3, 4)	6	მონეტები 1 და 6 არიან უფრო მძიმეები, ვიდრე მონეტა 4. 1 და 6 მონეტებს შორის უფრო მსუბუქია მონეტა 6.
getHeaviest(3, 5, 6)	5	მონეტა 5 არის უმძიმესი 3, 5 და 6 მონეტებში.
getMedian(1, 5, 6)	1	მონეტა 1 არის მედიანა 1, 5 და 6 მონეტებში.
getMedian(2, 4, 6)	6	მონეტა 6 არის მედიანა 2, 4 და 6 მონეტებში.
answer([3, 4, 6, 2, 1, 5])		პროგრამამ მოძებნა სწორი პასუხი მოცემული ტესტისათვის

სანიმუშო გრადერი

სანიმუშო გრადერს აქვს შემდეგი ფორმატი:

- სტრიქონი 1: T — ტესტების რაოდენობა
- თითოეულ სტრიქონში 2–დან $(T+1)$–მდე: მიმდევრობა 6 განსხვავებული რიცხვისაგან დიაპაზონში 1–დან 6–მდე: დალაგებული მონეტები მსუბუქიდან მძიმისაკენ.

მაგალითად, ორი ტესტისაგან შედგენილი შემავალი მონაცემები, სადაც მონეტების მიმდევრობაა 1 2 3 4 5 6 და 3 4 6 2 1 5 ასე გამოიყურება

```
2
1 2 3 4 5 6
3 4 6 2 1 5
```

გრადერი ბეჭდავს მასივს, რომელიც მიღებულია ფუნქცია `answer()`–ში პარამეტრის სახით.