Zadanie: KOL
Kolekcjoner Bajtemonów

Olimpiada Informatyczna

XXII OI, etap III, dzień drugi. Plik źródłowy `kol.*` Dostępna pamięć: 8 MB. *17.04.2015*

Bajtazar bardzo lubi kolekcjonować karty z Bajtemonami. Na każdej karcie w jego talii narysowany jest jeden Bajtemon wraz z numerem katalogowym, będącym liczbą całkowitą z przedziału $[1, 2 \cdot 10^9]$. Bajtazar nie zgromadził jeszcze wszystkich Bajtemonów. Jego kolekcja jest dość osobliwa: każdy Bajtemon, który się w niej znajduje, występuje na kilku kartach, co więcej, każdy na takiej samej liczbie kart.

Pewnego dnia Bajtazar zorientował się, że ktoś podkradł mu kilka (jedną lub więcej) kart z jego kolekcji. Wie, że na brakujących kartach był ten sam Bajtemon i że na szczęście został mu przynajmniej jeden egzemplarz tej karty. Niestety, nasz bohater ma bardzo mały rozumek i zdążył już zapomnieć, jaki to był Bajtemon. Czy jesteś w stanie mu pomóc i przypomnieć, jakich kart mu brakuje? Twój program też musi pamiętać o ograniczeniach pamięci...

Napisz program komunikujący się z biblioteką służącą do przeglądania talii kart Bajtazara, który znajdzie numer katalogowy podkradzionego Bajtemona.

Komunikacja

Aby użyć biblioteki, należy wpisać na początku programu:

- **C/C++**: `#include "ckollib.h"`
- **Pascal**: `uses pkollib;`

Biblioteka udostępnia następujące funkcje i procedury:

- `karta` – Daje w wyniku liczbę całkowitą z przedziału $[1, 2 \cdot 10^9]$ bądź 0. Wartość 0 oznacza koniec talii kart, zaś dodatnia liczba całkowita – numer katalogowy Bajtemona zapisany na kolejnej karcie. Po otrzymaniu wyniku funkcji 0, Twój program może wciąż wywoływać funkcję `karta` – odpowiada to kolejnemu przeglądaniu talii kart przez Bajtazara. Przy każdym przeglądaniu talii, numery katalogowe Bajtemonów podawane są w tej samej kolejności.
  - **C/C++:** `int karta();`
  - **Pascal:** `function karta: LongInt;`
- `odpowiedz(wynik)` – Odpowiada bibliotece, że `wynik` to numer katalogowy Bajtemona, który znajduje się na podkradzionych kartach. Wywołanie tej funkcji **kończy działanie Twojego programu**.
  - **C/C++:** `void odpowiedz(int wynik);`
  - **Pascal:** `procedure odpowiedz(wynik: LongInt);`

Twój program **nie może** czytać żadnych danych (ani ze standardowego wejścia, ani z plików). **Nie może** również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`) – pamiętaj jednak, że zużywa to cenny czas.

Ocenianie

Twój program otrzyma za dany test pełną punktację, jeśli przejrzy całą talię nie więcej niż raz (tj. nie wywoła funkcji `karta` po tym, gdy pierwszy raz da ona w wyniku wartość 0).

Jeśli program rozpocznie przeglądanie talii po raz drugi, otrzyma tylko 40% punktów za dany test.

W przypadku, gdy program zacznie przeglądać talię po raz trzeci, biblioteka przerwie działanie programu i nie uzyskasz za ten test żadnych punktów.

Liczba kart w talii nie przekroczy 60 000 000. Ponadto, w testach o łącznej wartości 20% punktów liczba kart w talii nie przekroczy 200 000.

Przykładowy przebieg programu

C/C++	Pascal	Wynik	Wyjaśnienie
`k = karta();`	`k := karta;`	13	Numer katalogowy Bajtemona z pierwszej karty.
`k = karta();`	`k := karta;`	13	
`k = karta();`	`k := karta;`	39	
`k = karta();`	`k := karta;`	26	
`k = karta();`	`k := karta;`	26	
`k = karta();`	`k := karta;`	0	Koniec talii.
`k = karta();`	`k := karta;`	13	Nowy obieg talii; karty są w tej samej kolejności, co poprzednio.
`odpowiedz(39);`	`odpowiedz(39);`		Udzielamy odpowiedzi. Program kończy swoje działanie.

Powyższy przebieg programu jest poprawny (dokonuje nie więcej niż dwóch przeglądnięć talii i daje poprawny wynik). Jednakże rozpoczyna on drugie przeglądanie talii, więc program otrzyma w tym przypadku 40% punktów. Aby uzyskać pełną punktację, należy dokonać tylko jednego przeglądnięcia talii.

Eksperymenty

W katalogu `dlazaw` dostępna jest przykładowa biblioteka, która pozwoli Ci przetestować poprawność formalną rozwiązania. Biblioteka wczytuje opis talii kart ze standardowego wejścia w następującym formacie:

- w pierwszym wierszu dodatnia liczba całkowita n – liczba kart w talii,
- w drugim wierszu n liczb całkowitych z przedziału $[1, 2 \cdot 10^9]$ – numery Bajtemonów na kolejnych kartach talii.

Przykładowa biblioteka **nie sprawdza**, czy faktycznie w talii istnieje tylko jeden Bajtemon narysowany na mniejszej liczbie kart niż wszystkie pozostałe Bajtemony występujące w talii.

Przykładowe wejście dla biblioteki znajduje się w pliku `kol0.in`. Po wywołaniu procedury `odpowiedz`, biblioteka wypisuje na standardowe wyjście informację o udzielonej odpowiedzi, liczbie wywołań funkcji `karta` i liczbie rozpoczętych przebiegów.

W tym samym katalogu znajdują się przykładowe rozwiązania `kol.c`, `kol.cpp` i `kol.pas` korzystające z biblioteki. Rozwiązania te nie są poprawne i zawsze odpowiadają, że szukanym numerem Bajtemona jest numer umieszczony na ostatniej karcie w talii.

Do kompilacji rozwiązania wraz z biblioteką służą polecenia:

- **C**: `gcc -O2 -static ckollib.c kol.c -lm`
- **C++**: `g++ -O2 -static ckollib.c kol.cpp -lm -std=gnu++0x`
- **Pascal**: `ppc386 -O2 -XS -Xt kol.pas`

Plik z rozwiązaniem i biblioteka powinny znajdować się w tym samym katalogu.