Zadanie Cukierki – LOGIA 18 (2017/18), etap 2

Treść zadania

Adam ma papierową torebkę. Wrzuca do niej cukierki, a następnie częstuje nimi znajomych. Cukierki są wrzucane i wyciągane pojedynczo. Postępuje tak wielokrotnie: wrzuca i częstuje. Adam zastanawia się, ile razy w torebce było dokładnie **n** cukierków. Napisz dwuparametrową funkcję **ilerazy**, której pierwszym parametrem jest **n** (od 1 do 10000). Drugim parametrem funkcji jest parzystej długości lista dodatnich liczb całkowitych, zawierająca na przemian liczby wrzucanych i wyjmowanych cukierków. Na przykład lista zawierająca 4, 1, 3 i 2 oznacza, że kolejno do torebki zostały wrzucone cztery cukierki, wyjęty jeden, wrzucone trzy i wyjęte dwa. Długość listy wynosi od 2 do 1000. Wynikiem funkcji jest liczba określająca, ile razy w torebce było dokładnie **n** cukierków.

Przykłady:

Python:	Logo:
wynikiem **ilerazy (2, [3, 2])** jest **2**,	wynikiem **ilerazy 2 [3 2]** jest **2**,
wynikiem **ilerazy (4, [5, 3, 5, 3, 2, 1])** jest **4**.	wynikiem **ilerazy 4 [5 3 5 3 2 1]** jest **4**.

W pierwszym przykładzie liczba cukierków w torebce wynosi kolejno: 0, 1, 2, 3, 2, 1.

W drugim przykładzie liczba cukierków w torebce wynosi kolejno: 0, 1, 2, 3, 4, 5, 4, 3, 2, 3, 4, 5, 6, 7, 6, 5, 4, 5, 6, 5.

Omówienie rozwiązania

Zadanie polega na policzeniu, ile razy określona liczba cukierków znajdowała się w torebce. W związku z tym potrzebna będzie zmienna, która będzie to pamiętać (jej wartość początkowa wynosi 0). W drugiej zmiennej będziemy pamiętać aktualną liczbę cukierków znajdujących się w torebce. Jej wartość początkowa także wynosi 0, ponieważ na początku torebka jest pusta. Zwróćmy uwagę, że mimo iż cukierki są wkładane i wyciągane pojedynczo, możemy jeden element listy potraktować całościowo. Jeśli przed włożeniem cukierków ich liczba w torebce była mniejsza niż **n**, a po włożeniu większa lub równa **n**, tzn., że w trakcie wkładania w torebce znajdowało się dokładnie **n** cukierków. A więc licznik należy powiększyć o jeden. Analogicznie, jeśli przed wyjmowaniem cukierków ich liczba w torebce była większa od **n**, a po wyjęciu mniejsza lub równa, to w trakcie wyjmowania znajdowało się dokładnie **n** cukierków. Za każdym razem modyfikujemy liczbę cukierków znajdujących się w torebce. Do rozwiązania zadania wystarczy więc pojedyncza pętla przeglądająca listę i w zależności od parzystości indeksu elementu wykonująca opisane działania.

Rozwiązanie w języku Python

```python
def ilerazy(n, lista):
    ilen = 0
    suma = 0
    for i in range(len(lista)):
        if i % 2 == 0:
            if suma < n and suma + lista[i] >= n:
                ilen += 1
            suma += lista[i]
        else:
            if suma > n and suma - lista[i] <= n:
                ilen += 1
            suma -= lista[i]
    return ilen
```

Elementy listy w Pythonie są indeksowane (numerowane) od 0, w związku z tym parzysty indeks oznacza wkładanie cukierków do torebki, a nieparzysty wyjmowanie.

Rozwiązanie w języku Logo

```
oto ilerazy :n :lista
  niech "ilen 0
  niech "suma 0
  powtórz długość :lista
    [
      jeżeli reszta npw 2 = 1
        [
          jeśli i :suma < :n (:suma + element npw :lista) >= :n [ zwiększ "ilen ]
          (zwiększ "suma element npw :lista)
        ]
        [
          jeśli i :suma > :n (:suma - element npw :lista) <= :n [ zwiększ "ilen ]
          (zmniejsz "suma element npw :lista)
        ]
    ]
  wy :ilen
już
```

Elementy listy w Logo są numerowane od jeden, w związku z tym nieparzysta wartość numeru powtórzenia pętli oznacza wkładanie cukierków do torebki, a parzysta wyjmowanie.

Testy

Testowanie rozwiązania warto rozpocząć od przykładów, których wyniki można prosto sprawdzić ręcznie np. podobnych, jak w treści zadania. Potem można testować działanie programu dla większych danych, choć i w tym przypadku warto tak dobrać dane, aby można było ocenić poprawność wyniku. W testach powinny być ujęte wszystkie możliwe przypadki, czyli: dodanie lub wyjęcie cukierków powoduje przejście przez wartość graniczną **n**, dodanie lub wyjęcie cukierków nie powoduje przejścia przez wartość graniczną.

Python	Logo	
ilerazy(1, [2, 1])	ilerazy 1 [2 1]	2
ilerazy(2, [3, 2, 1, 1, 2, 3])	ilerazy 2 [3 2 1 1 2 3]	5
ilerazy(100, [90, 80])	ilerazy 100 [90 80]	0
ilerazy(1, [2, 1, 2, 2, 3, 3])	ilerazy 1 [2 1 2 2 3 3]	4
ilerazy(2, [2, 1, 2, 1, 2, 1, 1, 2])	ilerazy 2 [2 1 2 1 2 1 1 2]	4
ilerazy(6, [3, 1, 3, 1, 3, 2, 3, 1])	ilerazy 6 [3 1 3 1 3 2 3 1]	3
ilerazy(51, [100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1])	ilerazy 51 [100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1]	99