PPPD - Lab. 11

Zadanie punktowane, lab 11, 2019/2020, autor: Maciej Bartoszuk

Temat: Zabawy z ciągami

Treść zadania

- W zadaniu **można** korzystać z metody `.append()`.
- **Nie można** korzystać zindeksowania ujemnego oraz `slice`.

Plik `dane.txt` zawiera ciąg liczb całkowitych, przy czym w każdym wierszu pliku znajduje się dokładnie jedna wartość.

Uwaga: wartości zapisane w pliku mogą być zarówno dodatnie jak i ujemne.

Zadania do wykonania

1. (2p) Napisz funkcję `wczytaj()`, która wczyta nieujemne wartości zapisane w pliku `dane.txt` i zwróci je w postaci listy liczbowej.

   W funkcji `main()` przy użyciu funkcji `wczytaj()` pobierz dane zawarte w pliku `dane.txt`.

2. (15p) Napisz funkcję `sprawdz_poprawnosc(y)`, która sprawdzi czy podany jej argument jest niepustą listą o elementach całkowitych i nieujemnych. Funkcja powinna zwrócić wartość `True` gdy wszystkie warunki są spełnione, oraz `False` w przeciwnym przypadku.

3. (2p) Napisz funkcję `podziel(y)`, która dla listy `y` składającej się z liczb całkowitych nieujemnych wyznaczy i zwróci listę zawierającą wszystkie kolejne podciągi rosnące listy `y`. Dokładniej, każdy element listy wynikowej powinien być listą zawierającą rosnący podciąg `y`-ka. Na przykład dla

   ```
   y == [1, 2, 1, 3, 4, 2, 4, 6, 8, 1, 0]
   ```

   powinniśmy otrzymac w wyniku listę postaci:

   ```
   [[1, 2], [1, 3, 4], [2, 4, 6, 8], [1], [0]]
   ```

   Wykorzystaj funkcję `sprawdz_poprawnosc()` w celu sprawdzenia czy podana jako argument lista `y` jest zgodna z założeniami funkcji. W przypadku gdy tak nie jest rzuć wyjątek.

   W funkcji `main()` wyznacz rosnące podciągi dla danych wczytanych z pliku `dane.txt`.

   Wskazówka: Zauważ, że dla dowolnej listy `t` wywołanie `t.append([])` spowoduje rozszerzenie listy `t` o jeden element będący pustą listą, np.

   ```
   t = [ [1, 2] ]
   t.append([])
   t == [ [1, 2], [] ]
   ```

4. (1p) Napisz funkcję `wypisz(x)`, która dla listy `x` określonej w poprzednim podpunkcie wypisze na konsoli wszystkie znalezione podciągi w następujący sposób:

```
z = podziel([1, 2, 1, 3, 4, 2, 4, 6, 8, 1, 0])
print(z)

# wynik:
x[ 0 ]:   1      2
x[ 1 ]:   1      3      4
x[ 2 ]:   2      4      6      8
x[ 3 ]:   1
x[ 4 ]:   0
```

W funkcji `main()` wypisz przy użyciu tej funkcji listę zwróconą przez funkcję `podziel()`.

5. (4p) Napisz funkcję `zlacz_posortowane(x)`. Funkcja przyjmuję jako argument `x` listę wyznaczoną przy użyciu funkcji `podziel()` i zwraca nową listę zawierającą wszystkie elementy z `x` uporządkowane rosnąco. Innymi słowy funkcja powinna dokonać sortowania przez złączenie poszczególnych elementów list z `x`-a tak by zachować porządek rosnący.

   Na przykład, w powyższym przykładzie będziemy mieć:

   ```
   [0, 1, 1, 1, 2, 2, 3, 4, 4, 6, 8]
   ```

   W funkcji `main()` wywołaj funkcję dla listy zwróconej przez funkcję `podziel()`i wypisz wynik.

Wynik działania programu:

```
y = wczytaj()
x = podziel(y)
wypisz(x)
# wynik wypisania:
x[ 0 ]:   1      2
x[ 1 ]:   1      3      4
x[ 2 ]:   2      4      6      8
x[ 3 ]:   2
x[ 4 ]:   2
x[ 5 ]:   1      5      9      10
x[ 6 ]:   1      2      4
x[ 7 ]:   2      7
x[ 8 ]:   4      9      11     18     20     22
x[ 9 ]:   1      3
x[10 ]:   2      3
x[11 ]:   1      7
x[12 ]:   1      10
x[13 ]:   0

print(zlacz_posortowane(z))
# wynik wypisania

[0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
 4, 4, 4, 4, 5, 6, 7, 7, 8, 9, 9, 10, 10, 11, 18, 20, 22]
```

Uwaga

- Jeśli program się nie kompiluje (interpretuje), ocena jest zmniejszana o połowę.
- Jeśli kod programu jest niskiej jakości (nieestetycznie formatowanie, mylące nazwy zmiennych itp.), ocena jest zmniejszana o 2 p.