



Opis

Pan Michał Meloman lubi wieczorami relaksować się przy dźwiękach muzyki ze swojej przepastnej kolekcji płyt. Jego kolekcja opatrzona jest plikami z metadanymi, zawierającymi informacje o różnych albumach – przykładowo rok wydania, nazwa zespołu oraz liczba utworów na albumie.

Twoim zadaniem jest napisanie programu który posortuje dane zebrane w formacie CSV (comma-separated values) po wybranej kolumnie. Twój program ma działać jak najszybciej i zużywać jak najmniej pamięci.

Wejście


Dane do programu wczytywane są ze standardowego wejścia. W pierwszej linii wejścia podawana jest dodatnia liczba naturalna z zakresu od 1 do 100. Oznacza ona liczbę zestawów danych które pojawią się następnie na wejściu.

Każdy zestaw zaczyna się od linii z trzema liczbami naturalnymi rozdzielonymi przecinkami. Pierwsza wyznacza liczbę wierszy w zestawie (nie licząc nagłówka). Druga to numer kolumny względem której należy posortować dane (numeracja zaczyna się od 1). Trzecia z liczb koduje porządek sortowania (1 oznacza sortowanie rosnące, a -1 malejące).

Każda kolumna zawiera wyłącznie jeden z dwóch typów danych - liczby naturalne lub napisy bez polskich znaków. Wartości w kolumnach są unikatowe. Sortowanie liczb naturalnych ma odbywać się zgodnie z naturalnym porządkiem, a napisów zgodnie z porządkiem słownikowym (zadany przez metodę „compareTo” klasy String). Dla danych występujących w pliku maksymalna długość napisu to 100 znaków, maksymalna długość liczby całkowitej w zapisie dziesiętnym to 5 znaków, a maksymalna liczba wierszy w zestawie to 10^5 .

Wyjście

Dla każdego zestawu danych program ma wypisać posortowane wiersze z danymi. Jako pierwsza od lewej ma zostać wyświetlona kolumna po której sortujemy, pozostałe kolumny bez zmienionej kolejności. Po każdym zestawie danych powinna pojawić się pojedyncza pusta linia.

	<p>Metody programowania 2022/2023</p> <p>Sortowanie metadanych</p>	<p>P_07</p>
---	--	-------------

Wymagania implementacyjne i inne uwagi

1. Do sortowania danych program ma wykorzystywać **algorytm QuickSort w wersji iteracyjnej**, o stałej złożoności pamięciowej oraz bez użycia stosu. Dla podzadań o rozmiarze mniejszym lub równym niż 5 należy wykorzystać algorytm SelectionSort. W efekcie średnia złożoność czasowa programu dla zestawu o rozmiarze N powinna wynosić **$O(N \log(N))$** .
2. Jediną biblioteką którą można zaimportować jest *java.util.Scanner*.
3. Program ma wczytywać dane do tablicy i ją modyfikować tak, aby uzyskać oczekiwany porządek (nie może wyłącznie wypisywać danych na ekran).
4. W pierwszej linii program powinien zawierać komentarz:


```
// Nazwisko i imię – nr grupy
```
5. Program powinien zawierać komentarze wyjaśniające działanie algorytmu oraz opisujące jego średnią złożoność wraz z uzasadnieniem. Dodatkowo załączyć należy przykładowe dane testowe.



Przykładowe wywołanie

Wejście

3

3,1,-1

Album,Year,Songs,Length

Stadium Arcadium,2006,28,122

Unlimited Love,2022,17,73

Californication,1999,15,56

3,2,1

Album,Year,Songs,Length

Stadium Arcadium,2006,28,122

Unlimited Love,2022,17,73

Californication,1999,15,56

3,4,-1

Album,Year,Songs,Length

Stadium Arcadium,2006,28,122

Unlimited Love,2022,17,73

Californication,1999,15,56

Wyjście

Album,Year,Songs,Length

Unlimited Love,2022,17,73

Stadium Arcadium,2006,28,122

Californication,1999,15,56

Year,Album,Songs,Length

1999,Californication,15,56

2006,Stadium Arcadium,28,122

2022,Unlimited Love,17,73

Length,Album,Year,Songs

122,Stadium Arcadium,2006,28

73,Unlimited Love,2022,17

56,Californication,1999,15