PPJ

I. Metoda printMonth przyjmuje jako argumenty int m i int y, opisujące odpowiednio miesiąc i rok. Uzupełnij ciało tej metody tak aby wyświetlała pełen miesiąc kalenda-rza, jak w przykładzie:

```
1                1   2   3   4   5
2   6   7    8   9  10  11  12
3  13  14   15  16  17  18  19
4  20  21   22  23  24  25  26
5  27  28   29  30  31
```

III. Stwórz klasę zawierającą funkcje statyczne operujące na tablicach typu int[]:

• static int[] add(int[] a, int elem) — przyjmuje posortowaną tablicę a i tworzy oraz zwraca tablicę o jeden element dłuższą niż a, w której dodano jeden element o wartości elem. Nowy element powinien zostać dodany w taki sposób, aby wynikowa tablica nadal była posortowana rosnąco. Tablica wejściowa a może mieć długość 0.
• static int[] delIndex(int[] a, int ind) — przyjmuje tablicę a i tworzy oraz zwraca tablicę o jeden element krótszą niż a, w której nie ma elementu o indeksie ind, który był w a.
• static int[] delFirst(int[] a, int e) — przyjmuje tablicę a i tworzy oraz zwraca tablicę o jeden element krótszą niż a, w której nie ma pierwszego elementu o wartości e. Jeśli nie istnieje taki element, zwracana jest niezmieniona tablica a.
• static int[] delLast(int[] a, int e) — przyjmuje tablicę a i tworzy oraz zwraca tablicę o jeden element krótszą niż a, w której nie ma ostatniego elementu o wartości e. Jeśli nie istnieje taki element, zwracana jest niezmieniona tablica a.
• static int[] delAll(int[] a, int e) — przyjmuje tablicę a i tworzy oraz zwraca krótszą tablicę, w której nie ma elementów o wartości e. Jeśli nie istnieją takie elementy, zwracana jest niezmieniona tablica a.
• static void info(int[] a) — przyjmuje tablicę a i wypisuje ją w jednej linii z informacją o jej rozmiarze.
• public static void swap(int[ ]tab,int source,int destination) - Uzupełnij ciało tej metody, tak aby wskazane przez parametry source i destination elementy tablicy zostały zamienione miejscami.

Uwagi:
• Nie używaj żadnych funkcji/klas spoza pakietu java.lang — w szczególności sortowania czy kolekcji.
• Nie kopiuj tablic ani ich fragmentów za pomocą pętli. Zamiast tego użyj statycznej metody arraycopy z klasy System (z pakietu java.lang, nie wymaga importów)
System.arraycopy(sArr, sIndex, tArr, tIndex, count)
która kopiuje count elementów tablicy sArr zaczynając od pozycji sIndex do tablicy tArr zaczynając od pozycji tIndex.
• W funkcji delIndex, wartość indeksu może być nieprawidłowa — nie musisz obsługiwać takiej sytuacji, po prostu pozwól, aby program uległ awarii.

```
public class ArrAddRemove {
	public static void main(String[] args) {
		int[] a = {};
		info(a); // line 1
		a = add(a, 4);
		a = add(a, 1);
		a = add(a, 3);
		a = add(a, 7);
		a = add(a, 4);
		a = add(a, 2);
		a = add(a, 7);
		a = add(a, 4);
		a = add(a, 8);
		a = add(a, 7);
		a = add(a, 4);
		a = add(a, 5);
		info(a); // line 2
		a = delIndex(a, 2);
		a = delLast(a, 7);
		a = delFirst(a, 7);
		info(a); // line 3
		a = delAll(a, 4);
		info(a); // line 4
	}
static int[] add(int[] a, int elem) {
// ...
}
static int[] delIndex(int[] a, int ind) {
// ...
}
static int[] delFirst(int[] a, int e) {
// ...
}
static int[] delLast(int[] a, int e) {
// ...
}
static int[] delAll(int[] a, int e) {
// ...
}
static void info(int[] a) {
// ...
}
}
```

Wyjście:

```
Length 0: [ ]
Length 12: [ 1 2 3 4 4 4 4 5 7 7 7 8 ]
Length 9: [ 1 2 4 4 4 4 5 7 8 ]
Length 5: [ 1 2 5 7 8 ]
```

III. Napisz funkcję **selSort**, która sortuje daną tablicę liczb całkowitych (uporządkowuje jej elementy od najmniejszego do największego) przy użyciu algorytmu sortowania przez wybieranie. Mianowicie, iterujemy po pozycjach (indeksach) tablicy od indeksu 0 do przedostatniego, a dla każdego indeksu (np. i) znajdujemy indeks najmniejszego elementu spośród tych o indeksach większych niż i (czyli po prawej stronie od i-tego elementu). Jeśli znaleziony element jest mniejszy niż i-ty, zamieniamy je miejscami. Zauważ, że liczba zamian będzie co najwyżej n - 1, gdzie n to rozmiar tablicy.
Napisz dwie funkcje statyczne, obie przyjmujące dwuwymiarową tablicę liczb całkowitych:

sortRows — sortuje osobno każdy "wiersz" tablicy (zauważ, że tablica nie musi być prostokątna);
sortCols — sortuje osobno każdą "kolumnę" tablicy (oczywiście możliwe jest tylko dla tablic prostokątnych).
W obu przypadkach możesz użyć wcześniej zdefiniowanej funkcji **selSort**, chociaż w drugim przypadku lepiej byłoby zaimplementować sortowanie wewnątrz funkcji **sortCols**.
Napisz również funkcję, która wypisuje dwuwymiarową tablicę na konsoli.
Na przykład, poniższy program:

```
public class Arr2DSelSort {
   public static void selSort(int[] arr) { ... }
   public static void sortRows(int[][] arr) { ... }
   public static void sortCols(int[][] arr) { ... }
   public static void printArr2D(int[][] arr) { ... }

   public static void main(String[] args) {
      int[][] a = {
         {3, 2, 6, 1, 3, 5, 6, 1, 3},
         {3, 1, 2, 1, 5, 7, 2},
         {8, 9, 2, 1}
      };
      System.out.println("Before:");
      printArr2D(a);
      sortRows(a);
      System.out.println("After:");
      printArr2D(a);

      int[][] b = {
         {3, 2, 6, 1, 6},
         {7, 1, 2, 1, 5},
         {5, 3, 1, 8, 7},
         {8, 9, 2, 7, 1}
      };
      System.out.println("Now columns\nBefore:");
      printArr2D(b);
      sortCols(b);
      System.out.println("After:");
      printArr2D(b);
   }
}
```

powinien wydrukować:
[3, 2, 6, 1, 3, 5, 6, 1, 3]
[3, 1, 2, 1, 5, 7, 2]
[8, 9, 2, 1]
After:
[1, 1, 2, 3, 3, 3, 5, 6, 6]
[1, 1, 2, 2, 3, 5, 7]
[1, 2, 8, 9]
Now columns
Before:
[3, 2, 6, 1, 6]
[7, 1, 2, 1, 5]
[5, 3, 1, 8, 7]
[8, 9, 2, 7, 1]
After:
[3, 1, 1, 1, 1]
[5, 2, 2, 1, 5]
[7, 3, 2, 7, 6]
[8, 9, 6, 8, 7]

IV. W programie poniżej:

```
public class Arr2DJagged {
   public static void main(String[] args) {
      int[][] arr = {
         {1, 3},
         {3, 4, 5, 8},
         {6, 8},
         {9}
      };
      double[] res = getAverages(arr);
      for (double e : res) System.out.print(e + " ");
   }
   // ...
}
```

dodaj statyczną funkcję getAverages, która:
przyjmuje dwuwymiarową tablicę liczb całkowitych (niekoniecznie prostokątną);
zwraca tablicę liczb typu double, o rozmiarze równym liczbie "wierszy" przekazanej do funkcji dwuwymiarowej tablicy. Kolejne elementy wynikowej tablicy powinny być równe średnim arytmetycznym wszystkich elementów kolejnych "wierszy" tablicy wejściowej.
Dla tablicy jak powyżej, wynik powinien być:

```
2.0 5.0 7.0 9.0
```

V. Dana jest dwuwymiarowa tablica jak w poniższym kodzie:

```
int tab[ ] [ ] = {
{ 1 , 2 , 3 , 4} ,
{ 5 , 6 , 7 , 8} ,
{ 9 ,10 ,11 ,12}
,{13 ,14 ,15 ,16}
```

};
Utwórz program, który odczyta wartości idąc po spirali tak aby otrzymać następujący efekt:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

VI. Gracz rzuca dwoma kostkami do gry, uzyskując dwie niezależne liczby z przedziału 1–6. Wynik losowania przechowaj w dwuelementowej tablicy zmiennych typu int.Wykorzystując tekstowy interfejs komunikacji z użytkownikiem, pozwól na wprowadzenie komend:

- (s)how - wyświetli wszystkie wyniki losowania;
- (r)oll - wykona kolejny rzut kośćmi, wynik którego zostanie dopisany do tablicy przechowującej tablice z wynikami dotychczasowych losowań;