Polsemestrálny test
praktická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Zdrojový kód pre prvé dve úlohy nájdete na stránke, resp. v moodli. Evaluátor vyžaduje tieto názvy tried a tiež názvy metód, ktoré sú zadané. Všetky ostatné názvy premenných, vnorených tried a ostatných pomocných metód si môžete upraviť.

1. Priemer časov (5b) Marek si plánuje bežeckú sezónu na celý rok. Na kratšie trate mu stačia bežné tréningy, ale pri dlhších pretekoch musí venovať príprave viac úsilia. Aby sa vyhol zraneniam a zvládol náročnejšie trate, vždy pred nimi zaradí intenzívnejší tréning – beh na dvojnásobnú vzdialenosť.
Ak je nasledujúci pretek náročný (jeho dĺžka dosahuje zadanú hranicu), Marek si musí do plánu poznačiť, že pred ním absolvuje prípravný beh na dvojnásobnú vzdialenosť. Pomôžte mu zostaviť tréningový plán tak, aby vedel, kedy sa má intenzívne pripravovať.

Do triedy `SpajanyZoznam` pridajte metódu `seasonPlan`, ktorá modifikuje spájaný zoznam tak, že pred každú hodnotu prekračujúcu zadanú hranicu vloží jej dvojnásobok.

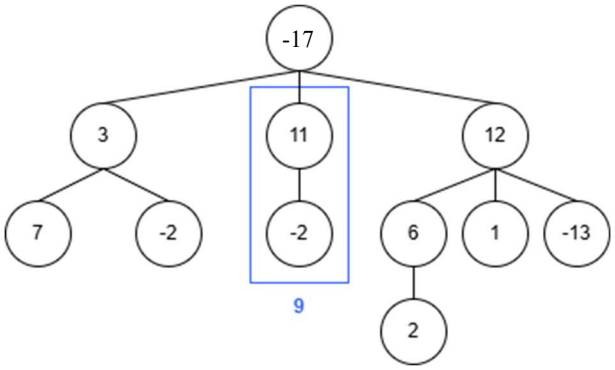
```
public void seasonPlan(int hardThreshold)
```

Metóda nech pracuje v lineárnom čase vzhľadom k dĺžke zoznamu, t.j. $O(n)$, kde n je dĺžka zoznamu. Za opakovaný prechod spájaným zoznamom je bodový zisk znížený o 1 bod. Metóda nevytvára nový zoznam, ale modifikuje aktuálny.

Metóda `seasonPlan` pridá do plánu pretekov [10, 37, 12, 42, 35] intenzívne tréningy nasledovne:

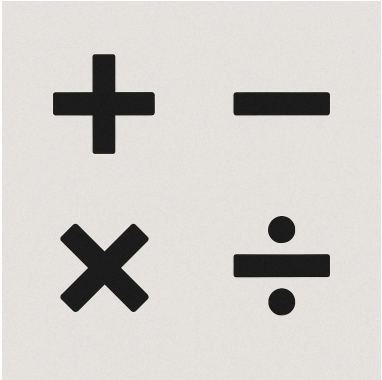
- `seasonPlan(30)` → [10, 74, **37**, 12, 84, **42**, 70, **35**]
- `seasonPlan(40)` → [10, 37, 12, 84, **42**, 35]
- `seasonPlan(50)` → [10, 37, 12, 42, 35]
- `seasonPlan(10)` → [20, **10**, 74, **37**, 24, **12**, 84, **42**, 70, **35**]

2. Hľadanie podstromu s maximálnym súčtom (5b) Predstavte si, že máte strom, ktorý reprezentuje sieť výdavkov a príjmov v oddeleniach nejakej firmy. Každý uzol v tomto strome predstavuje určité finančné hodnoty – môžu byť kladné, ak ide o príjmy, alebo záporné, ak ide o výdavky. Strom zobrazuje hierarchiu, kde každý uzol môže mať viacero podriadených uzlov (oddelenia), ktoré ovplyvňujú celkový finančný obraz podniku.



Uvažujme triedu Oddelenie, ktorá uchováva hodnotu samotného oddelenia a zoznam podriadených oddelení. Vašou úlohou je analyzovať tento strom a nájsť podstrom s najvyšším súčtom hodnôt, ktorý reprezentuje najbohatšiu časť podniku. Podstrom je definovaný ako **uzol a všetci jeho potomkovia**, teda všetko, čo je pod týmto uzlom vrátane neho samotného. Cieľom je určiť zisk najziskovejšieho podstromu, t.j. najvyšší súčet všetkých uzlov, ktoré sa v ňom nachádzajú.

```
public int richestDepartment()
```



3. Výraz so zadaným výsledkom (5b) Počas prenosu dôležitého matematického výrazu sa stratili všetky operátory. Zostala len neprázdna postupnosť nenulových čísel a výsledok, ktorý by mal vzniknúť ich správnym spojením. Vašou úlohou je zistiť, či existuje spôsob, ako medzi čísla vložiť operácie (+, -, *, /), aby sme dostali požadovaný výsledok.
Pri výpočte platia nasledovné pravidlá:

- Operácie sa vykonávajú postupne zľava doprava, bez prednosti násobenia a delenia.
- Operácia delenia je celočíselné delenie.

V triede `Matematika` vytvorte metódu `findOperators`, ktorá vráti, či je možné vložiť matematické operátory medzi každé dve čísla tak, aby sa výraz rovnal zadanej hodnote result.

```
public boolean findOperators(int[] numbers, int result)
```

Prílkady:

- `findOperators`([10, 5, 4, 2], 58) -> **`true`**, pretože 10+5*4-2=58
- `findOperators`([2, 2, 3, 4], 28) -> **`true`**, pretože 2*2+3*4=28 alebo 2+2+3*4=28
- `findOperators`([2, 4, 7], 2) -> **`false`**, pretože neexistujú vhodné operátory
- `findOperators`([1, 1], 3) -> **`false`**, pretože neexistujú operátory, ktorých výsledok by bola 3