DÚ 4

10 riadkov kódu

Vašou úlohou bude dopísať do templatu 10 lambda funkcií tak, aby všetky testy boli splnené. Z tohto vyplýva niekoľko pravidiel:

- Každá správna lambda funkcia je za 1 bod. Ak test odhalí chybu, tak je to automaticky 0 bodov za príslušnú časť.
- Ak program neočakávané skončí, tak celé riešenie je hodnotené počtom bodov 0.
- Ak sa program nedá skompilovať príkazom `javac Main.java FDS/*java`, tak riešenie je hodnotené počtom bodov 0 (príkaz `java --version` vracia "openjdk 17.0.5 2022-10-18", alebo kompatibilný variant).
- Je zakázané meniť súborovú štruktúru.
- Je zakázané meniť obsah súborov mimo špecifikovaných nižšie častí.
- Je zakázané používať kľúčové slovo import, return, kučeravé zátvorky, vytvárať anonymné/lokálne triedy (neberte to ako súťaž v SQL injections alebo hru Untrusted, tieto úlohy sa dajú napísať bez akýchkoľvek "hackov" na jeden riadok každú, aj bez využitia for-cyklov a prúdov).
- Lambda funkcie, ktoré máte doplniť, sa nachádzajú v subore FDS/Part1.java, v rozhraní Part1. Meniť je povolené každý výskyt reťazca "null" (bez úvodzoviek) v zdrojovom kóde rozhrania Part1. Všetko ostatné má ostať bez zmeny.

Perzistentné zoznamy

Zoznamy celých čísel budeme reprezentovať pomocou triedy FDS.FL (viď súbor FDS/FL.java). Tu je jej kompletný predpis:

```
public class FL {
   private final Integer head;
   private final FL tail;

   public FL(Integer head, FL tail) {
       this.head = head;
       this.tail = tail;
   }

   public Integer head() {return head;}
   public FL tail() {return tail;}
}
```

Prázdny zoznam `[]` je reprezentovaný hodnotou `null`. Jednoprvkový zoznam `[a]` je reprezentovaný ako `FL(a, null)`, dvojprvkový zoznam `[a, b]` --- ako `FL(a, FL(b, null))` atď. Prístup ku nultému prvku zoznamu sa robí pomocou metódy head, t.j. `[a,b,c].head() == a`. Metóda tail vráti zoznam bez nultého prvku, t.j. `[a,b,c].tail() == [b, c]`.

Prvky zoznamu číslujeme od 0. Nultý prvok zoznamu A vieme teda dostať ako `A.head()`, prvý prvok ako `A.tail().head()`, druhý ako `A.tail().tail().head()` atď.

U: (0 bodov) Skúste vytvoriť nejaký zoznam pomocou tejto triedy.

Bodované úlohy (1 bod za každú)

1. isEmpty(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala true, ak je zoznam A prázdny
2. size(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala dĺžku vstupného zoznamu.
3. min(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala minimum spomedzi prvkov zoznamu A, respektíve null, ak je prázdny.
4. push_back(A, x): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala zoznam A + [x] (t.j. za posledný prvok zoznamu A sa vloží nový prvok x).
5. pop_back(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala zoznam, obsahujúci všetky prvky zoznamu A až na posledný v rovnakom poradí.
6. reverse(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala zoznam prvkov zoznamu A v opačnom poradí.
7. concat(A1, A2): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala zoznam, pozostávajúci z prvkov A1 a následne prvkov A2. Formálne, concat(A1, A2)[i] = A1[i] pre i od 0 po size(A1)-1 a zároveň concat(A1, A2)[size(A1) + i] = A2[i] pre všetky i od 0 po size(A2)-1 (výraz A[i] znamená "i-tý prvok zoznamu A").
8. contains(A, n): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala true, ak zoznam A obsahuje prvok n, a false v opačnom prípade.
9. insert(A, n): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala vzostupne utriedený zoznam prvkov zoznamu concat(A, [n]), pričom môžete predpokladať, že vstupný zoznam A je vzostupne utriedený.
10. sort(A): Doplňte text do templatu v súlade s pravidlami tak, aby táto funkcia vracala vzostupne utriedený zoznam prvkov zoznamu A.

Kostra rozhrania Part1 vyzerá následovne:

```
public interface Part1 {
   Function<FL, Boolean> isEmpty = fl ->
           null;
   Function<FL, Integer> size = fl ->
           null;
   Function<FL, FL> reverse = fl ->
           null;
   Function<FL, Function<Integer, FL>> push_back = fl -> n ->
           null;
   Function<FL, FL> pop_back = fl ->
           null;
   Function<FL, Function<FL, FL>> concat = fl1 -> fl2 ->
```

```
            null;
    Function<FL, Function<Integer, Boolean>> contains = fl -> n ->
            null;
    Function<FL, Integer> min = fl ->
            null;
    Function<FL, Function<Integer, FL>> insert = fl -> n ->
            null;
    Function<FL, FL> sort = fl ->
            null;
}
```