Zaawansowane programowanie obiektowe
Lab. 2

1. (1.5 pkt) Zaimplementuj funkcję
   double LevQWERTY(String s1, String s2),
   która zwraca ważoną odległość Levenshteina między napisami s1 i s2, gdzie wagi zależne są od wzajemnego położenia pary znaków na klawiaturze.
   Konkretniej, odl. Levenshteina bazuje na 3 elementarnych operacjach: wstawienia znaku (ang. *insertion*), usunięcia znaku (ang. *deletion*) oraz zastąpienia znaku innym (ang. *substitution*). W naszym przypadku waga operacji insercji i delecji ma wynosić 1, natomiast waga substytucji wynosi:
   - 0.5, jeśli odnośna para znaków sąsiaduje w rzędzie na klawiaturze,
   - 1, w przeciwnym przypadku.

   Zakładamy, że s1 i s2 mogą zawierać tylko małe litery łacińskie oraz spacje.

   Przykłady:
   LevQWERTY("kot", "kita") == 1.5 (1 insercja (a) + 1 substytucja znaków sąsiadujących w rzędzie (o <--> i)).
   LevQWERTY("kwiat", "kwist") == 0.5 (1 substytucja znaków sąsiadujących w rzędzie (a <--> s)).
   LevQWERTY("drab", "dal") == 2 (1 delecja (r) + 1 substytucja znaków niesąsiadujących w rzędzie (b <--> l)).

   Napisz testy z użyciem jUnit sprawdzające poprawność napisanej funkcji.

   Wskazówka: zastosuj tablicę asocjacyjną z małymi literami łacińskimi jako kluczami oraz zbiorami liter z nimi sąsiadujących jako wartościami.

   Formuła programowania dynamicznego dla obliczania odl. Levenshteina + przykład:
   https://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Levenshteina

2. (2 pkt) Narysuj w konsoli, z wykorzystaniem rekurencji pionową linijkę o zadanych parametrach: (długość w danych jednostkach, liczba poziomów zagnieżdżeń).
   Udokumentuj swoją funkcję i obejrzyj wygenerowanego (-> javadoc) HTML-a.
   https://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

   Przykład, (2, 4):

```
---- 0
-
--
-
---
-
--
-
---- 1
-
--
-
---
-
--
-
---- 2
```