Lista 8

Uwaga: Materiały do tej listy znajdują się w pliku `https://math.uni.wroc.pl/~jagiella/files/p1python/lista8.zip`. Jeśli otwieranie niektórych plików tekstowych sprawia problemy, funkcję `open` można wywołać z parametrem opcjonalnym `encoding='utf8'`.

Zadanie 1 (1 punkt). *Słownikiem zapotrzebowań* nazwiemy słownik, w którym kluczami są napisy (nazwy potrzeb), a z każdym jest stowarzyszony jeden z trzech napisów `"low"`, `"medium"`, lub `"high"` oznaczających poziomy potrzeb (przy czym brak klucza oznacza brak odpowiedniej potrzeby). Przykłady:

```
{"drink": "low", "fun": "low"}  # (1)
{"food": "low", "drink": "low", "fun": "medium"}  # (2)
{"food": "medium", "oxygen": "high"}
```

Dla słowników `d1` i `d2` takiej postaci powiemy, że zapotrzebowania `d1` są nie większe, niż zapotrzebowania `d2` jeśli poziom każdej potrzeby w `d1` jest nie większy, niż poziom tej potrzeby w `d2`. Przykładowo, zapotrzebowania słownika z przykładu (1) są nie większe, niż zapotrzebowania (2).

Napisz funkcję, która dla pary takich słowników `d1`, `d2` sprawdza, czy zapotrzebowania `d1` są nie większe, niż zapotrzebowania `d2`.

Zadanie 2 (1,25 punktu). Napisz funkcję `rewrite_redacted(input_file, output_file, word)`, która przepisuje zawartość pliku tekstowego (o nazwie) `input_file` do pliku `output_file`, zastępujac wszystkie wystąpienia słowa `word` napisem `"[REDACTED]"`, niezależnie wielkości liter w słowie. Dla uproszczenia załóż, że po każdym słowie może wystąpić co najwyżej jeden znak interpunkcyjny spośród `,.!?` i że słowa w każdej linijce oddzielone są dokładnie jedną spacją.

Zadanie 3 (1,75 punktu). Plansza pewnej gry składa się z komnat. Niektóre komnaty są połączone (dwukierunkowymi) korytarzami. Każda komnata ma unikalną nazwę (napis niezawierający przecinków) i może zawierać skarb. W materiałach do listy znajdują się pliki `board.txt` i `chambers.csv`. Pierwszy plik zawiera dane o korytarzach łączących komnaty. W każdej jego linijce znajdują się nazwy komnat oddzielone przecinkami. Pierwsza z komnat wymienionych w linijce jest połączona korytarzami z każdą pozostałą wymienioną w tej linijce. Przykładowe linijki:

```
Sypialnia,Sala tronowa,Studium
Studium,Sypialnia,Sala tronowa,Biblioteka
```

Drugi plik opisuje obecność skarbów w każdej z komnat. Każda jego linijka składa się z (oddzielonych przecinkami) nazwy komnaty oraz liczby (0 lub 1). Komnata o podanej nazwie zawiera skarb wtedy i tylko wtedy, gdy liczba wynosi 1. Przykładowe linijki:

```
Biblioteka,1
Stajnia,0
```

Komnaty są *sąsiadami* gdy są połączone korytarzem. Napisz następujące funkcje, działające na podstawie danych wczytanych z obu plików:

- `num_close_treasures(chamber)` – dla komnaty o nazwie `chamber` zwraca liczbę jej sąsiednich komnat zawierających skarb.
- `get_nearby_treasures(chamber)` – dla komnaty o nazwie `chamber` zwraca listę komnat, które są sąsiadami sąsiadów `chamber` i zawierają skarb.
- `competing_chambers(chamber1, chamber2)` – dla komnat o nazwie `chamber1` i `chamber2` zwraca `True`, gdy obie komnaty mają wspólną sąsiednią komnatę zawierającą skarb (i `False` w przeciwnym przypadku).

Uwaga: Możesz założyć, że `board.txt` zawiera informację o wszystkich obecnych korytarzach i wszystkich komnatach. Podobnie, `chambers.csv` opisuje zawartość każdej z komnat (niekoniecznie w takiej kolejności, jak kolejność występowania komnat w `board.txt`).