Wstęp do programowania
Pracownia 10
Data publikacji: 18.12.2024

Uwaga: Przedświąteczna wprawka powinna być łatwa i przyjemna.

Premia za tę listę wynosi 0.5, przyznawana jest osobom, które zdobyły co najmniej 2p za zadania z tej listy. Maksimum wynosi 4p. (tradycyjnie, dwa pierwsze zadania są łatwe).

Zadanie 1.(1pkt) Szyfrowanie metodą Cezara polega na tym, że każdą literę danego słowa zamienia się na literę przesuniętą o k pozycji (zgodnie z porządkiem alfabetycznym, w którym po ostatniej literze (ż) następuje litera pierwsza (a)). W szyfrze Cezara kluczem umożliwiającym szyfrowanie (i odszyfrowanie) jest liczba k.

Napisz funkcję `ceasar(s,k)`, która dla danego słowa `s` i klucza `k` znajduje wartość szyfrogramu. Pamiętaj, by używać polskiego alfabetu (aąbcćdeęfghijklłmnńoóprsśtuwyzźż). Zastanów się, jak można byłoby tu wykorzystać słowniki i funkcję `zip` do utworzenia zwięźlejszego i eleganckiego kodu.

Zadanie 2.(1pkt) Parę słów nazwiemy *parą cesarską* (a występujące w niej słowa *cesarskimi*), jeżeli są one wzajemnie swoimi szyfrogramami w szyfrze Cezara (tzn. każde z nich otrzymujemy z drugiego za pomocą odpowiedniego przesunięcia wszystkich liter; oczywiście przesunięcie powinno być nietrywialne, czyli nie może być identycznością). Napisz program, który znajduje najdłuższe polskie słowo cesarskie (jeżeli więcej niż jedno osiąga maksymalną długość powinieneś wypisać je wszystkie).

Zadanie 3.(1pkt) Łamigłówką arytmetyczną jest zadanie, w którym należy literom przyporządkować (różne) cyfry w ten sposób, by będące treścią zadania dodawianie było prawdziwe. Przykładowe zadania to:

```
  SEND          CIACHO
+ MORE        +  CIACHO
-------       ---------
 MONEY          NADWAGA
```

Napisz program, który rozwiązuje łamigłówki arytmetyczne. W programie powinna być funkcja, której argumentem jest napis przedstawiający zagadkę (przykładowo `"send + more = money"`, a wynikiem słownik kodujący (jakieś) rozwiązanie. Gdy rozwiązanie nie istnieje, funkcja powinna zwracać pusty słownik (ew. wartość `None`).

Zadanie 4.(1pkt) Napisz dwie funkcje wykorzystujące rekurencję (lub jedną za połowę punktów). W obu definicjach powinieneś skorzystać z mechanizmu *list comprehension*, postaraj się, by definicje były możliwie jak najbardziej zwięzłe.

a) Napisz rekurencyjną funkcję, która generuje zbiór wszystkich sum podzbiorów listy liczb `L` (czyli jeżeli `L` była równa `[1,2,3,100]`, to funkcja powinna zwrócić zbiór

```
set([0,1,2,3,4,5,6, 100, 101, 102, 103, 104, 105, 106])
```

b) Napisz rekurencyjną funkcję, która generuje wszystkie ciągi niemalejące o długości N, zawierające liczby od A do B.

Zadanie 5.(1pkt)($\star$) Napisz funkcję, która dla zbioru elementów zwraca listę wszystkich relacji równoważności[1] określonych na tych elementach. Relacje równoważności będziemy wyrażać, jako listę zbiorów (klas abstrakcji tej relacji). Zadbaj o to, by w wyniku nie powtarzały się żadne relacje. Twoja funkcja dla zbioru `{1 ,2}` powinna zwrócić listę: `[ [{1}, {2}], [ {1,2} ] ]` (lub inną, w której kolejność elementów w którejś liście jest różna)

Zadanie 6.(1pkt)($\star$) Jesteś pisarzem literatury fantastycznej (raczej użytkowej, niż artystycznej, szczerze mówiąc). Całkiem dobrze sobie z tym radzisz, ale masz kłopot z wymyślaniem imion dla bohaterów. To zadanie ma być użytecznym narzędziem rozwiązującym taki problem, czyli wspomagającym twórczy proces wymyślania imion (nazwisk) dla tego typu literatury. Należy rozwiązać je w następujący sposób:

- Imię będziemy losować znak po znaku.
- Jak zobaczysz, wygodnie przyjąć, że każde imię zaczyna się od pary znaków `^^` a kończy znakiem `$` (oczywiście można tu wybrać inne oznaczenia, nie powinieneś również tych dziwnych znaków pokazywać użytkownikowi)
- Imię powinno mieć pewną długość minimalną, przykładowo 4 znaki, powinieneś też określić długość maksymalną.
- Prawdopodobieństwo wylosowania znaku na pozycji `i` powinno zależeć od znaków `i-1` oraz `i-2`.

[1] Równoważnie można myśleć o podziałach tego zbioru, czyli o listach rozłącznych zbiorów dających w sumie zbiór początkowy

- Prawdopodobieństwa te powinieneś szacować przeglądając plik z rzeczywistymi imionami (podany na kno, możesz skorzystać z innego – na przykład jednoznacznie słowiańskiego, jeżeli uda Ci się taki odnaleźć). Przykładowo, gdyby jedynymi imionami były Paweł i Ewelina, wówczas dla znaków `we` możliwe byłyby tylko dwie kontynuacje, mianowicie `ł` oraz `l`, każda z prawdopodobieństwem $\frac{1}{2}$.

Twój program powinien wczytać listę imion, oszacować na jej podstawie prawdopodobieństwo losowania znaków, następnie wylosować kilkanaście imion zgodnych z powyżej naszkicowanymi zasadami. Jeżeli nie do końca wiesz, jak się zabrać za to zadanie, śmiało pytaj prowadzącego na zajęciach lub na wykładzie. W przypadku wspomnianych dwóch imion, oprócz nich moglibyśmy wylosować imiona *Pawelina* i *Eweł*