Wstęp do programowania
Pracownia 12

Uwaga: Wszystkie zadania z tej listy będzie można oddawać do końca semestru. Do maksimum wliczają się 3 punkty. O grafach będzie na wykładzie 13

Premia za tę listę wynosi 0.5, przyznawana jest osobom, które zdobyły co najmniej 2p za zadania z tej listy.

Zadanie 1.(1pkt) W zadaniu zastanowimy się, jak zamienić **wodę** w **wino**. A dokładniej, rozważamy następującą łamigłówkę, w której ruchem jest zmiana jednej litery w czteroliterowym słowie polskim (użyj tego samego zbioru co w poprzednich zadaniach) w taki sposób, żeby powstało inne czteroliterowe polskie słowo. Czynności należy powtarzać, aż otrzyma się docelowe słowo. Przykładowy ciąg wyrazów zamieniających **mąka** w **keks**:
`['mąka', 'mika', 'miks', 'kiks', 'keks']`
Napisz funkcję, która dla pary słów znajduje najkrótszą ścieżkę zamieniającą jedno w drugie (zwracaną jako listę słów) lub listę pustą, jeżeli ścieżka nie istnieje. Sprawdź, jaką wartość zwraca ona dla pary woda i wino. Uwaga: jeżeli nie wiesz, jak się do tego zabrać, poczekaj na wykład 13.

Zadanie 2.(1pkt) Zadanie z wilkiem, kozą i kapustą zdefiniowane jest następująco:

- Na jednej stronie rzeki znajduje się łódź (z przewoźnikiem), wilk, koza i kapusta.
- Ani wilk, ani koza, ani kapusta nie umieją same prowadzić łodzi, a przewoźnik jest w stanie zabrać na raz co najwyżej jedno z nich.
- Jeżeli w którymś momencie na brzegu będzie sam wilk z kozą, albo koza z kapustą (bez przewoźnika), wówczas nastąpi niepożądana konsumpcja.
- Celem jest doprowadzenie do sytuacji w której cała czwórka bezpiecznie znajdzie się na drugim brzegu.

Napisz program, który rozwiązuje to zadanie (czyli wypisuje sekwencję ruchów/stanów prowadzących do rozwiązania) na dwa sposoby:

1. Wykonując przeszukiwanie w głąb
2. Wykonując losowe dozwolone ruchy.

Zadanie 3.(1pkt) Popraw klasę `Set` z wykładu 12, by obsługiwała również takie konstrukcje jak `len(s)` oraz `s1 & s2`, `s1 - s2`, `s1 | s2` (ta ostatnia była na wykładzie, ale mocno nieidealnie)

Uwaga: częścia tego zadania jest samodzielne znalezienie, jaki nazwy muszą mieć `__specjalne_metody__` (*dunder methods*) przeciążające operatory.

Zadanie 4.(1pkt) Szyfr przestawieniowy to taki szyfr, w którym każdej literce z polskiego alfabetu przypisana jest inna literka (konsekwentnie, w ramach całego komunikatu). W tym i kolejnym zadaniu, będziemy łamać takie szyfry (czyli pisać programy, które znajdują komunikat, w sytuacji, gdy mamy znamy jedynie szyfrogram). Będziemy zakładać, że słowa w szyfrogramie oddzielone są spacjami i (dla zwiększenia czytelności komunikatu), między nimi czasami znajdują się znaki interpunkcyjne (niezaszyfrowane, otoczone spacjami). Zakładamy również, że wszystkie słowa w komunikacie występują w słowniku (z polskimi słowami z jednej z poprzednich list) i że nie mamy żadnych dodatkowych informacji o języku (np. o częstościach liter, czy wyrazów).

Napisz program, który umie rozszyfrować dwa pierwsze szyfrogramy ze SKOS-u. Uwaga: w obu tych szyfrogramach wszystkie słowa mają unikalną permutacyjną postać normalną (to znaczy, że znajomość tejże postaci pozwala jednoznacznie wybrać słowo). Uwaga2: każdy szyfrogram jest w osobnym wierszu, każdy był też szyfrowany osobną permutacją.

Zadanie 5.(0.5pkt) Zmodyfikuj program z zadania 2, by poradził sobie z jeszcze jakimś przykładem, w którym nie wszystkie słowa mają unikalną permutacyjną postać normalną (na przykład trzeci szyfrogram ma tę własność, będąc zarazem bardzo łatwym do odszyfrowania)

Zadanie 6.(1pkt) Zmodyfikuj program, by poradził sobie z wszystkimi przykładami ze SKOS-u. Uwaga: to już trochę trudniejsze zadanie. W rozwiązaniu można wykorzystać fakt, że nawet nieunikalna normalna postać permutacyjna daje pewne wskazówki o możliwych przyporządkowaniach liter. Szyfrogram 'ółółó' mówi na przykład, że literze 'ó' nie można przypisać litery 'ą' (bo żadne słowo nie zaczyna się na 'ą'). Można też z niego wydedukować inne rzeczy, patrząc na wszystkie słowa o postaci permutacyjnej 1-2-1-2.

Szyfrogramy ze SKOS-u da się odszyfrować w czasie kilku sekund każdy (a niektóre w ułamki sekundy). Ale jest to zadanie trochę trudniejsze.

Zadanie 7.(1pkt) Dodaj do programu z obracającymi się kwadratami ruch ze stałą, losowaną na początku prędkością i zmianę koloru każdego kwadratu w zależności od położenia (w dowolny sposób). Ruch powinien być z zawijaniem, czyli obiekt wychodzący z ekranu po prawej stronie powinien pojawić się na lewym skraju (i analogicznie we wszystkich innych kierunkach)

Uwaga: Twoje rozwiązanie musi być inne niż rozwiązanie zadania z animacjami z poprzedniej listy.

Zadanie 8.(1+2pkt) Napisz, używając biblioteki turtle Wielki System Przesuwania Prostokątów. Powinien on obsługiwać następujące zadania:

1. Losować pewną liczbę kolorowych prostokątów umieszczanych na ekranie
2. Obsługiwać mechanizm przenoszenia prostokątów (drag and drop). Metodę onclick przedstawiliśmy na wykładzie, mogą się przydać również onrelease i ondrag.
3. Zmieniać jakoś (na żądanie użytkownika) kolejność prostokątów, żeby nie tracić dostępu do małych prostokątów pokrytych przez duże

Zadanie 9.(1pkt) Zdefiniuj klasę R (ułamki), która implementuje operacje na ułamkach (na wykładzie 13 dowiesz się, jak można tworzyć 'własne' liczby). Klasa powinna mieć następujące właściwości.

1. Dodawać, odejmować, dzielić i mnożyć w sposób dokładny ułamki.
2. Używać do tych operacji naturalnej składni pythona (czyli operatorów `+*-/`, operatorów `+=`, `-=`, etc
3. Wypisywać ułamki (w taki sposób, by działała również instrukcja `print [u1,u2,u3]`, gdzie `u1,u2,u3` są ułamkami.
4. Porównywać ułamki.

Oczywiście nie możesz korzytać z klasy Fraction z biblioteki standardowej.

Zadanie 10.(1pkt) Popraw efektywność tej klasy za pomocą skracania ułamków. Wykorzystaj ją do znalezienia odpowiedzi na pytanie, jaka 5-cyfrowa liczba pierwsza znajduje się najwcześniej w rozwinięciu dziesiętnym liczby e. Czy wiesz, jaka anegdota stoi za tym zadaniem?