Wstęp do programowania 2014
Pracownia 10

Uwaga: Na tej liście znowu będą wprawki o tematyce wybranej przez prowadzącego ćwiczenia. Podczas tych zajęć można oddawać zadania z listy 8 za 0.5 i 9 za 1.

Premia za tę listę wynosi 0.5, przyznawana jest osobom, które zdobyły co najmniej 1.5p za zadania z tej listy. Maksimum dla tej listy wynosi 3.5p.

Zadanie 1.(1,⋆pkt) Rozwiąż dowolne zadanie z poprzednich list, którego do tej pory nie zrobiłeś. Możesz też nic nie rozwiązywać, tylko wskazać jakieś zadanie za które, ze względu na spóźnienie, dostałeś pół punktu – wówczas dostaniesz brakującą połowę.

Zadanie 2.(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 3.(2pkt) 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 4.(1pkt) Liczba e jest (jak wiadomo) sumą następującego szeregu:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

Napisz funkcję sumującą K początkowych wyrazów tego szeregu na dwa sposoby:

a) Wykonując obliczenia na liczbach zmiennoprzecinkowych (czyli sumując wyrazy: `1.0/silnia(n)`)

b) Wykonując obliczenia na liczbach wymiernych, na końcu przekształcanych do liczby zmiennoprzecinkowej. W Pythonie liczb wymiernych nie ma, ale są długie liczby całkowite, które umożliwiają dokładne obliczenia na liczbach wymiernych. Przekształcenie do postaci zmiennoprzecinkowej można zrobić mnożąc licznik przez `10 ** k` (gdzie k jest liczbą cyfr po przecinku, która nas interesuje[1]), a następnie wykonując dzielenie całkowite (i wstawiając tam gdzie trzeba kropkę).

Przyjmijmy, że prawidłowa wartość e to `math.exp(1)`. Pokaż, jak zmienia się różnica między prawidłową wartością e, a jego przybliżeniami liczonymi wg obu metod. Uwaga: użyty tu ciąg jest trochę 'zbyt porządny', zapewne efekt byłby wyraźniejszy, gdybyśmy użyli innego ciągu. Zastanów się jakiego i sprawdź tę hipotezę (to jest część bonusowa, warta dodatkowe 0.5p).

[1] Aby wybrać k, należy w konsoli pythonowej wpisać na przykład math.pi