Wstęp do programowania
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 2p za zadania z tej listy. Maksimum wynosi 4p.

Zadanie 1.(1pkt) Zadanie to kontynuuje zadanie z grą w życie. Będziemy rozważać bitewny wariant tej gry, w której walczą dwie kolonie komórek, Niebieska i Czerwona. Początkowo, każda z nich w całości mieści się na kwadracie o wymiarze $K \times K$ (roboczo przyjmujemy K = 15), cała plansza jest prostokątem, składającym się z lewej (niebieskiej) i prawej (czerwonej) części (plansza jest „zawinięta"). Obowiązują te same zasady dotyczące umierania komórek, co w oryginalnej grze w życie. Jeśli chodzi o urodziny, to komórki rodzą się (jak w oryginale), gdy mają 3 sąsiadów, a kolor takiej komórki jest taki, jak kolor większości rodziców.

Stan początkowy gry (dla obu drużyn), zawarty jest plikach tekstowych (`niebieski.txt` oraz `czerwony.txt`[1]. Plik z wejściem składa się z K wierszy, każdy po K znaków. Puste pola oznaczamy znakiem kropki, z zapełnione znakiem `#`.

Preferowane jest przedstawianie gry w postaci graficznej. Zakładamy, że gra kończy się w dwóch przypadkach:

1. Co najmniej jedna kolonia nie ma już komórek.
2. Pewien stan gry się powtórzył (to znaczy, że nie wydarzy się już nic nowego).

Wynik gry zależy od liczby żywych komórek obu graczy w momencie zakończenia gry.

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.(1.5pkt) 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) Wybierz jedno zadanie z Analizy Literackiej (pojawią się na SKOS) i zaprezentuj jego rozwiązanie.

[1] Powinieneś przygotować te pliki, lub, co może być zabawniejsze, przygotować jeden, a drugi wziąć od koleżanki lub kolegi.