Zestaw 2

1. Mamy zagnieżdżoną listę, która może zawierać różne heterogeniczne typy, na przykład inną listę, ale również krotkę, słownik. Dodaj element o kolejnej wartości w **najbardziej zagnieżdżonej** liście. Napisz program, który zrobi to uniwersalnie, dla dowolnego zagnieżdżenia, również jeśli pojawią się inne typy. Dla [1 [2, 3] 4] chodzi o [1 [2, 3, **4**] 4], dla [3, 4, [2, [1, 2, [7, 8], 3, 4], 3, 4], 5, 6, 7] powinno być [3, 4, [2, [1, 2, [7, 8, **9**], 3, 4], 3, 4], 5, 6, 7]. Jeżeli największe zagnieżdżenie na danym poziomie się powtórzy, należy dodać w obu zagnieżdżeniach, czyli dla [1, [3], [2]] należy uzyskać [1, [3, **4**], [2, **3**]]. Przykład bardziej złożony: list1 = [1, 2, [3, 4, [5, {'klucz': [5, 6], 'tekst': [1, 2]}], 5], 'hello', 3, [4, 5], (5, (6, (1, [7, 8])))]. Tutaj na takim samym, największym poziomie zagnieżdżenia, są listy będące wartościami w słowniku (listy [5, 6], [1, 2]) a także zagnieżdżona w krotkach (lista [7, 8]) i do to do nich powinien zostać dodany kolejny element. Zatem oczekiwane jest: [1, 2, [3, 4, [5, {'klucz': [5, 6, **7**], 'tekst': [1, 2, **3**]}], 5], 'hello', 3, [4, 5, 6], (5, [6, [7, 8, **9**]])].

2. Dla dowolnego podanego łańcucha znakowego wypisać: ile jest w nim *słów* (poprzez słowo rozumiemy ciąg co najmniej jednego znaku innego niż znak przestankowy, dla uproszczenia przyjmijmy, że liczymy a-z, A-Z i 0-9 jako coś, co składa się na słowa), ile *liter*, ile *cyfr*, oraz wypisać statystykę częstości występowania poszczególnych liter oraz cyfr.

3. Napisać program konwertujący liczby zapisane w systemie rzymskim (wielkimi literami I, V, X, L, C, D, M) na liczby arabskie w zakresie liczb 1-3999, i odwrotnie. Proszę kontrolować poprawność danych wejściowy, również w formacie rzymskim. Proszę spróbować napisać najbardziej kompaktowy (krótki) kod.

4. Mamy trzy liczby całkowite, x, y, z reprezentujące wymiary prostopadłościanu, oraz pewną liczbę naturalną n. Wypisz listę wszystkich możliwych współrzędnych (i, j, k) na trójwymiarowej siatce, gdzie i+j+k nie jest równe n. Warunki: $0 \leq i \leq x$, $0 \leq j \leq y$, $0 \leq k \leq z$. Rozwiązanie zapisz w postaci list składanych (list comprehesion), ale można zacząć od zagnieżdżonych pętli. Przykład. Niech x = 1, y = 1, z = 2, n = 3. Lista wszystkich permutacji trójek [i, j, k] w tym przykładzie: [[0,0,0], [0,0,1], [0,0,2], [0,1,0], [0,1,1], [0,1,2], [1,0,0], [1,0,1], [1,0,2], [1,1,0], [1,1,1], [1,1,2]]. Elementy, które nie sumują się do 3 to: [[0,0,0], [0,0,1], [0,0,2], [0,1,0], [0,1,1], [1,0,0], [1,0,1], [1,1,0], [1,1,2]]. Parametry x, y, z, n wczytać na początku za pomocą funkcji input().

5. Mamy liczbę naturalną N w zapisie binarnym (czyli składa się tylko z 0 i 1). Binarna przerwa to sekwencja zer otoczonych z lewej i prawej strony 1. Na przykład liczba 9 (decymalnie) binarnie wynosi 1001 i posiada jedną binarną przerwę długości 2. Liczba 529 ma binarną reprezentację 1000010001, zatem ma dwie binarne przerwy, o długości 4 i 3. Liczba 20 ma reprezentację 10100 zawiera zatem jedną binarną przerwę o długości 1. Liczba 15 ma reprezentację 1111, a zatem żadnej binarnej przerwy. Napisz funkcję:

```
def fun(N)
```

która dla podanej liczby naturalnej N (uwaga: liczby w systemie dziesiętnym) zwraca długość jej najdłuższej binarnej przerwy, albo 0, jeśli nie ma ani jednej przerwy. Na przykład, dla N = 1041, które binarnie jest 10000010001, ma najdłuższą przerwę binarną 5. Należy przyjąć, że argument N może być z przedziału [1..2147483647]. Wskazówka: warto skorzystać z operatora przesunięcia bitowego >>. Można podejrzeć jak wygląda liczba w zapisie binarnym poprzez rzutowanie bin(N).