Завдання №1

1. **УВАГА!** Завдання цього заняття передбачають написання функцій та, відповідно, вставку фрагменту коду замість цілої програми. Ваш код не повинен містити жодних інших операції для введення або виведення.
2. Кожна функція має сприймати довільні значення, які задовольняють описаному в завданні формату.
3. Формат введення даних та виведення або повернення результатів має ТОЧНО збігатися із вказаним.

1) **Розробити функцію** clean_list(list_to_clean),
яка приймає 1 аргумент -- список будь-яких значень (рядків, цілих та дійсних чисел) довільної довжини,
та повертає список, який складається з тих самих значень, але не містить повторів елементів. Це значить, що у випадку наявності значення в початковому списку в кількох екземплярах перший "екземпляр" значення залишається на своєму місці, а другий, третій та ін. видаляються.

Наприклад
Виклик функції: clean_list([1, 1.0, '1', -1, 1])
Повертає: [1, 1.0, '1', -1]
Виклик функції: clean_list(['qwe', 'reg', 'qwe', 'REG'])
Повертає: ['qwe', 'reg', 'REG']
Виклик функції: clean_list([32, 32.1, 32.0, -123])
Повертає: [32, 32.1, 32.0, -123]
Виклик функції: clean_list([1, 2, 1, 1, 3, 4, 5, 4, 6, '2', 7, 8, 9, 0, 1, 2, 3, 4, 5])
Повертає: [1, 2, 3, 4, 5, 6, '2', 7, 8, 9, 0]

2) **Розробити функцію** counter(a, b),
яка приймає 2 аргументи -- цілі невід'ємні числа a та b,
та повертає число -- кількість різних цифр числа b, які містяться у числі a.

Наприклад
Виклик функції: counter(12345, 567)
Повертає: 1
Виклик функції: counter(1233211, 12128)
Повертає: 2

Виклик функції: counter(123, 45)
Повертає: 0

3) **Розробити функцію** super_fibonacci(n, m),
яка приймає 2 аргументи -- додатні цілі числа n та m (0 < n, m <= 35),
та повертає число -- n-й елемент послідовності супер-Фібоначчі порядку m.

Нагадуємо, що послідовність Фібоначчі -- це послідовність чисел, в якій кожний елемент дорівнює сумі двох попередніх. Послідовністю супер-Фібоначчі порядку m будемо вважати таку послідовність чисел, в якій кожний елемент дорівнює сумі m попередніх. Перші m елементів (з порядковими номерами від 1 до m) вважатимемо одиницями.

Наприклад
Виклик функції: super_fibonacci(2, 1)
Повертає: 1
Виклик функції: super_fibonacci(3, 5)
Повертає: 1
Виклик функції: super_fibonacci(8, 2)
Повертає: 21
Виклик функції: super_fibonacci(9, 3)
Повертає: 57

4) **Розробити функцію** file_search(folder, filename),
яка приймає 2 аргументи -- список folder та рядок filename,
та повертає рядок -- повний шлях до файлу або папки filename в структурі folder.

Файлова структура folder задається наступним чином:

Список -- це папка з файлами, його 0-й елемент містить назву папки, а всі інші можуть представляти або файли в цій папці (1 файл = 1 рядок-елемент списку), або вкладені папки, які так само представляються списками. Як і в файловій системі вашого комп'ютера, шлях до файлу складається з імен всіх папок, в яких він міститься, в порядку вкладеності (починаючи з зовнішньої і до папки, в якій безпосередньо знаходиться файл), розділених "/".

Вважати, що імена всіх файлів є унікальними. Повернути логічне значення False, якщо файл не знайдено.

Наприклад
Виклик функції: file_search(['C:', 'backup.log', 'ideas.txt'], 'ideas.txt')
Повертає: 'C:/ideas.txt'
Виклик функції: file_search(['D:', ['recycle bin'], ['tmp', ['old'], ['new folder1', 'asd.txt', 'asd.bak', 'find.me.bak']], 'hey.py'], 'find.me')
Повертає: False
Виклик функції: file_search(['/home', ['user1'], ['user2', ['my pictures'], ['desktop', 'not this', 'and not this', ['new folder', 'hereiam.py']]], 'work.ovpn', 'prometheus.7z', ['user3', ['temp'],], 'hey.py'], 'hereiam.py')
Повертає: '/home/user2/desktop/new folder/hereiam.py'

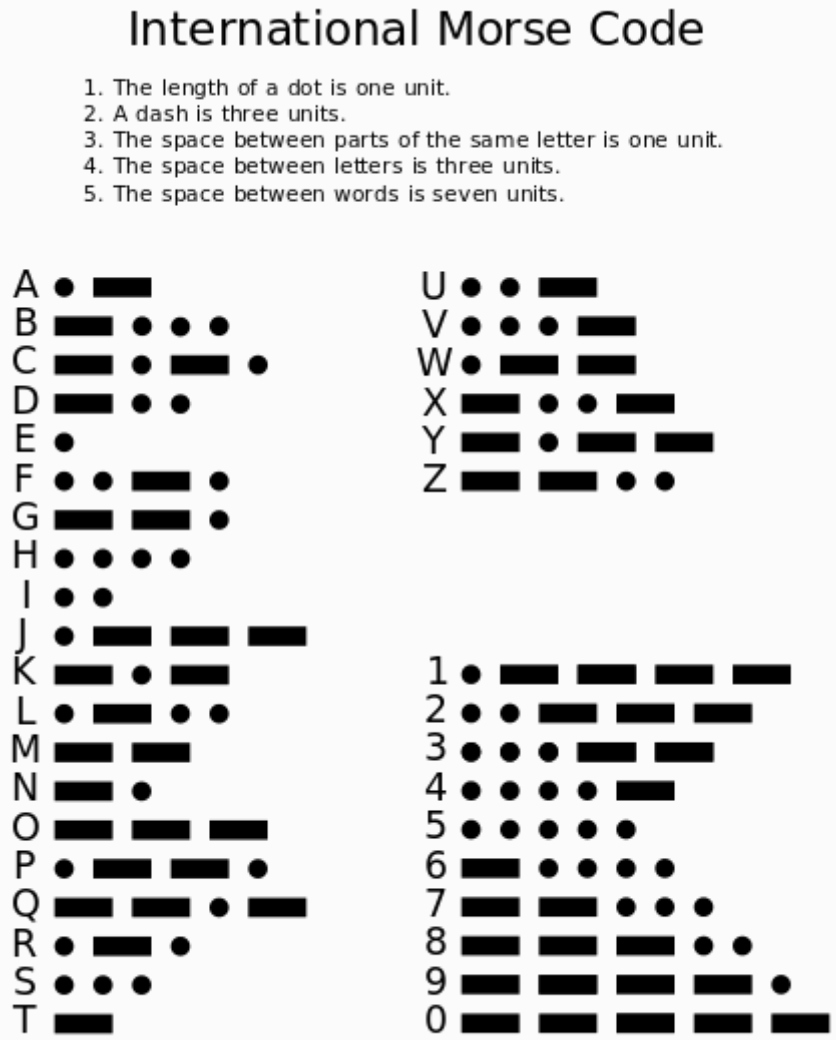
Лапки не повертаються і використані тут для розрізнення логічного False та рядків.

5)Розробити функцію count_holes(n),
яка приймає 1 аргумент -- ціле число або рядок, який містить ціле число, та повертає ціле число -- кількість "отворів" у десятковому записі цього числа друкованими цифрами (вважати, що у "4" та у "0" по одному отвору), або рядок ERROR, якщо переданий аргумент не задовольняє вимогам: є дійсним або взагалі не числом.

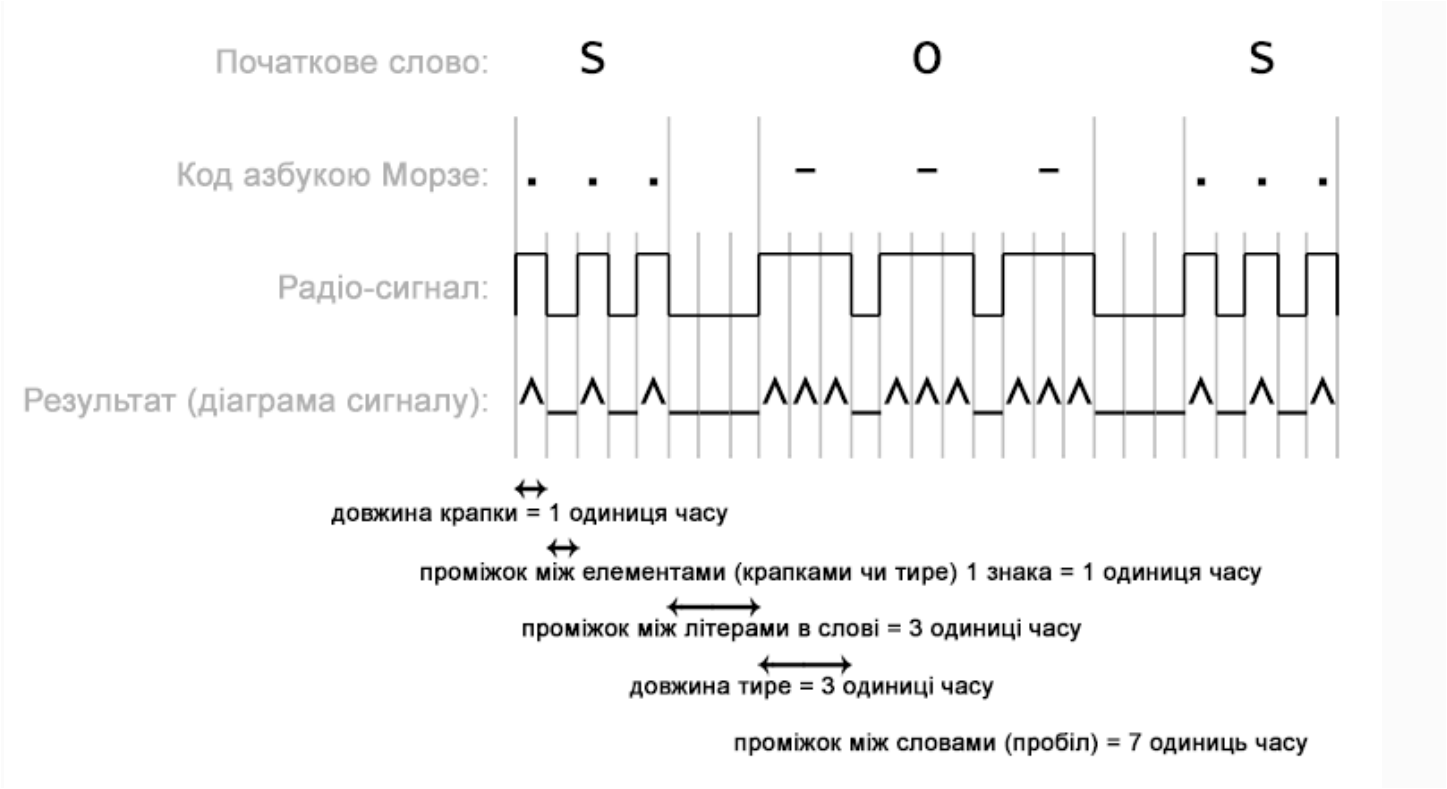
Незначущими нулями на початку запису числа, якщо такі є, нехтувати.

Наприклад
Виклик функціїї: count_holes('123')
Повертає: 0
Виклик функціїї: count_holes(906)
Повертає: 3
Виклик функціїї: count_holes('001')
Повертає: 0
Виклик функціїї: count_holes(-8)
Повертає: 2
Виклик функціїї: count_holes(-8.0)
Повертає: ERROR

6) **Розробити функцію** encode_morze(text),
яка приймає 1 аргумент -- рядок,
та повертає рядок, який містить діаграму сигналу, що відповідає переданому тексту, закодованому міжнародним кодом Морзе для англійського алфавіту. Розділові та інші знаки, що не входять до латинського алфавіту, ігнорувати. Регістром літер нехтувати.



Для передачі повідомлення за одиницю часу приймається тривалість однієї крапки. Тривалість тире дорівнює трьом крапкам. Пауза між елементами одного знака -- одна крапка, між знаками в слові -- 3 крапки, між словами -- 7 крапок. Словом вважати послідовність символів, обмежена пробілами. Результуюча "діаграма" демонструє наявність сигналу в кожний проміжок часу: на і-й позиції знаходиться "^", якщо в цей момент передається сигнал, і "_", якщо сигналу немає. Зайві паузи в кінці повідомлення на "діаграмі" відсутні.



Наприклад

Виклик функції: encode_morze('Morze code')

Повертає:

^^^_^^^___^^^_^^^_^^^___^_^^^_^___^^^_^^^_^_^___^_____
__^^^_^_^^^_^___^^^_^^^_^^^___^^^_^_^___^

Виклик функції: encode_morze('Prometheus')

Повертає:

^_^^^_^^^_^___^_^^^_^___^^^_^^^_^^^___^^^_^^^___^___^^
^___^_^_^_^___^___^_^_^^^___^_^_^

Виклик функції: encode_morze('SOS')

Повертає: ^_^_^___^^^_^^^_^^^___^_^_^

Виклик функції: encode_morze('1')

Повертає:

```
morse_code = {
   "A" : ".-",
   "B" : "-...",
   "C" : "-.-.",
   "D" : "-..",
   "E" : ".",
   "F" : "..-.",
   "G" : "--.",
   "H" : "....",
```

```
  "I" : "..",
  "J" : ".---",
  "K" : "-.-",
  "L" : ".-..",
  "M" : "--",
  "N" : "-.",
  "O" : "---",
  "P" : ".--.",
  "Q" : "--.-",
  "R" : ".-.",
  "S" : "...",
  "T" : "-",
  "U" : "..-",
  "V" : "...-",
  "W" : ".--",
  "X" : "-..-",
  "Y" : "-.--",
  "Z" : "--.."
}
```

7) **Розробити функцію** saddle_point(matrix),
яка приймає 1 аргумент -- прямокутну матрицю цілих чисел, задану у вигляді списка списків,
та повертає тьюпл із координатами "сідлової точки" переданої матриці або логічну константу False, якщо такої точки не існує.

Сідловою точкою вважається такий елемент матриці, який є мінімальним (строго менше за інші) у своєму рядку та максимальним (строго більше за інші) у своєму стовпці, наприклад:
матриця:
1 2 3
0 2 1
"1" в лівому верхньому кутку (за координатами (0;0)) є сідловою точкою матриці.

Вважати, що передані дані є коректними, тобто матриця не містить інших значень крім цілих чисел, а всі вкладені списки мають однакову довжину.
Результуючий тьюпл містить два числа -- порядкові номери сідлової точки в

рядку (індекс його списка у зовнішньому списку) та в стовпці (індекс у внутрішньому списку).

Наприклад
1 2 3
3 2 1
Виклик функції: saddle_point([[1,2,3],[3,2,1]])
Повертає: False
8 3 0 1 2 3 4 8 1 2 3
3 2 1 2 3 9 4 7 9 2 3
7 6 0 1 3 5 2 3 4 1 1
Виклик функції:
saddle_point([[8,3,0,1,2,3,4,8,1,2,3],[3,2,1,2,3,9,4,7,9,2,3],[7,6,0,1,3,5,2,3,4,1,1]])
Повертає: (1, 2)
21
Виклик функції: saddle_point([[21]])
Повертає: (0, 0)

8) **Розробити функцію** find_most_frequent(text),
яка приймає 1 аргумент -- текст довільної довжини, який може містити літери латинського алфавіту, пробіли та розділові знаки (,.:;!?-);
та повертає список слів (у нижньому регістрі), які зустрічаються в тексті найчастіше.

Слова, записані через дефіс, вважати двома словами (наприклад, "hand-made"). Слова у різних відмінках, числах та з іншими перетвореннями (наприклад, "page" та "pages") вважаються різними словами. Регістр слів -- навпаки, не має значення: слова "page" та "Page" вважаються 1 словом.

Якщо слів, які зустрічаються найчастіше, декілька -- вивести їх в алфавітному порядку.

Наприклад
Виклик функції: find_most_frequent('Hello,Hello, my dear!')
Повертає: ['hello']
Виклик функції: find_most_frequent('to understand recursion you need first to understand recursion...')
Повертає: ['recursion', 'to', 'understand']

Виклик функції: find_most_frequent('Mom! Mom! Are you sleeping?!!!')
Повертає: ['mom']

9)Розробити функцію convert_n_to_m(x, n, m),
яка приймає 3 аргументи -- ціле число (в системі числення з основою n) або рядок x, що представляє таке число, та цілі числа n та m (1 <= n, m <= 36),
та повертає рядок -- представлення числа x у системі числення m.

У випадку, якщо аргумент x не є числом або рядком, або не може бути представленням цілого невід'ємного числа в системі числення з основою n, повернути логічну константу False.

В системах числення з основою більше десяти для позначення розрядів із значенням більше 9 використовувати літери латинського алфавіту у верхньому регістрі від A до Z. У вхідному x можуть використовуватися обидва регістри.

Вважати, що в одиничній системі числення число записується відповідною кількістю нулів.

Наприклад
Виклик функції: convert_n_to_m([123], 4, 3)
Повертає: False
Виклик функції: convert_n_to_m("0123", 5, 6)
Повертає: 102
Виклик функції: convert_n_to_m("123", 3, 5)
Повертає: False
Виклик функції: convert_n_to_m(123, 4, 1)
Повертає: 000000000000000000000000000
Виклик функції: convert_n_to_m(-123.0, 11, 16)
Повертає: False
Виклик функції: convert_n_to_m("A1Z", 36, 16)
Повертає: 32E7