Завдання №1

Завдання:

Розробити додаток для конвертації між одиницями відстані з підтримкою метричної та імперської систем вимірювання. Співвідношення для конвертації ви можете взяти з таблиці. Початково додаток повинен розпізнавати метри (*m*), сантиметри (*cm*), дюйми (*in*) та фути (*ft*), і підтримувати конвертацію між будь-якими з цих одиниць виміру.

Також необхідно реалізувати можливість розширювати список одиниць, що підтримуються шляхом задання правил конвертації за допомогою окремого файлу. Формат файлу - на ваш розсуд. Для прикладу, розширте ваш додаток, додавши в файл значення для міліметрів (*mm*), ярдів (*yd*) та кілометрів (*km*).

Вхідні параметри:

Об'єкт, що містить відстань, задану для конвертації (distance) зі значенням (value) та шкалою (unit), а також позначення одиниці виміру для шкали, в яку повинна бути зроблена конвертація (convert_to), наприклад:

```
{"distance": {"unit": "m", "value": 0.5}, "convert_to": "ft"}
```

Вихідні дані:

Об'єкт, що містить отримане значення відстані, округлене до сотих, а також позначення відповідної одиниці виміру, наприклад:

```
{"unit": "ft", "value": 1.64}
```

Завдання №2

Завдання:

Розробити простий додаток для сортування і відбору даних за заздалегідь заданими правилами. Додаток повинен уміти працювати зі списками об'єктів довільної структури, відбирати об'єкти, котрі містять ключі з відповідними значеннями, а також сортувати об'єкти за значенням, використовуючи природний порядок сортування.

Наприклад, якщо для даних виду:

```
{"data": [{"name": "John", "email": "john2@mail.com"},
          {"name": "John", "email": "john1@mail.com"},
          {"name": "Jane", "email": "jane@mail.com"}]}
```

задати умову:

```
{"condition": {"include": [{"name": "John"}], "sort_by": ["email"]}}
```

що містить два правила - *include* і *sort_by* (де правило *include* приймає набір пар ключ:значення для перевірки записів на відповідність, а правило *sort_by* приймає набір ключів для сортування), результатом будуть об'єкти, що містять лише записи з ім'ям *John* та відсортовані по ключу *email*:

```
{"result": [{"name": "John", "email": "john1@mail.com"},
            {"name": "John", "email": "john2@mail.com"}]}
```

Плануючи підхід до дизайну коду додатка, необхідно передбачити можливість розширення функціонала шляхом додавання у код нових “модулів” із правилами. Важливо, щоб усі модулі мали між собою ідентичну структуру, були ізольовані один від одного та іншого коду додатка та взаємодіяли з основним кодом за єдиним принципом. Для прикладу, ви можете додати новий модуль з правилом *exclude,* яке буде відкидати записи, що містять ключі з певним значенням.

Вхідні параметри:

Об'єкт зі списком даних (*data*) та умовою для обробки (*condition*):

```
{"data": [{"user": "mike@mail.com", "rating": 20, "disabled": false},
          {"user": "greg@mail.com", "rating": 14, "disabled": false},
          {"user": "john@mail.com", "rating": 25, "disabled": true}],
 "condition": {"exclude": [{"disabled": true}], "sort_by": ["rating"]}}
```

Вихідні дані:

Об'єкт з даними, отриманими після застосування умови обробки (*result*):

```
{"result": [{"user": "greg@mail.com", "rating": 14, "disabled": false},
            {"user": "mike@mail.com", "rating": 20, "disabled": false}]}
```

Завдання №3 ‡

Завдання:

Дослідники дізналися, що в деякій частині космосу є нерухомо розташований астероїд з унікальним мінералом. Для його точного знаходження був розроблений новий тип простих одноразових зондів, які під час активації один раз визначають точну відстань від себе до астероїда.

Необхідно написати функцію, яка задаватиме координати активації зондам і, отримуючи відстані від кожного з них до астероїда, відшукає координати астероїда, витративши при цьому найменшу кількість зондів.

Частина космосу, в якій розташований астероїд, представлена кубом зі стороною ребра 100. Для вибору координат астероїда необхідно написати функцію, яка задасть випадкову точку в цьому кубі з цілими координатами: a(x, y, z).

Також необхідно написати окрему функцію f, яка, приймаючи координати зонда z(x, y, z), поверне відстань між цією точкою та астероїдом. Наприклад, для позиції астероїду a(0, 0, 10) та координат зонда z(0, 0, 0) результатом роботи функції буде:

```
f(z) = 10  // відстань між z(0, 0, 0) та a(0, 0, 10) дорівнює 10
```

Вхідні параметри:

—

Вихідні дані:

Як результат - програма повинна повернути координати астероїда *(asteroidLocation)*, кількість використаних зондів (zonds_spent) та їх координати *(zondCoordinates)*:

```
{"result": {
  "asteroidLocation": {"x":34, "y": 50, "z": 60},
  "zondCoordinates": [{"x": 10, "y": 9, "z": 21}, ..., {"x": 10, "y": 4, "z": 11}],
  "zondsSpent": 44
}}
```

‡ *Ви можете вирішити тільки одне з завдань: або №3, або №4, вирішення обох завдань буде плюсом*

Завдання №4 ‡

Завдання:

Необхідно реалізувати опитування, в якому порядок та список запитань залежить від переданої конфігурації. Опитування має підтримувати лише запитання з варіантами відповідей, наприклад:

```
{"What is your marital status?": ["Single", "Married"]}
{"Are you planning on getting married next year?": ["Yes", "No"]}
{"How long have you been married?": ["Less than a year", "More than a year"]}
{"Have you celebrated your one year anniversary?": ["Yes", "No"]}
```

Запитання в опитуванні повинні визначатися динамічно на основі відповідей користувача - наступне запитання має залежати від відповіді на попереднє. Вам необхідно продумати, як буде працювати ця логіка, і розробити формат конфігурації (він буде відрізнятись від прикладу вище). Дана конфігурація дозволить задавати правила, що пов'яжуть запитання з відповідями.

Для тестування роботи опитування потрібно створити скрипт, що працює з кодом логіки опитування і проходить по всіх можливих шляхах опитувань. Для вирішення завдання не потрібно реалізовувати інтерфейс користувача, достатньо імплементувати скрипт та логіку опитування.

Вхідні параметри:

Конфігурація, у вибраному вами форматі з запитаннями та доступними відповідями, що пов’язує відповіді користувача та наступні запитання.

Вихідні дані:

Об’єкт, що є результатом роботи скрипту тестування, з інформацією про кількість усіх можливих шляхів опитувань (*paths.number*), та всіма можливими послідовностями запитань з відповідями (*paths.list*):

```
{paths: {number: 3, list: [
    [{"What is your marital status?": "Single"},
     {"Are you planning on getting married next year?": "Yes/No"}],
    [{"What is your marital status?": "Married"},
     {"How long have you been married?": "Less than a year"}],
    [{"What is your marital status?": "Married"},
     {"How long have you been married?": "More than a year"},
     {"Have you celebrated your one year anniversary?": "Yes/No"}],

]}}
```

‡ *Ви можете вирішити тільки одне з завдань: або №3, або №4, вирішення обох завдань буде плюсом*

Примітки до виконання завдань

Під час написання додатків зверніть увагу на наступне:

- усі завдання повинні бути вирішені БЕЗ використання бібліотек чи фреймворків, наприклад таких як: React, Angular, Vue.js, Express
- файли додатків, що містять JavaScript або TypeScript код, обов'язково повинні мати відповідне розширення **.js*, або **.ts*
- фінальне рішення може бути орієнтоване як на запуск в браузері, так і в Node.js середовищі - на ваш розсуд

- код додатків необхідно розбити на логічні блоки так, щоб він був компактним, легкочитним та не містив повторень
- додатки повинні коректно реагувати на широкий спектр можливих вхідних значень та обробляти виняткові ситуації
- всі завдання мають бути вирішені оптимальним чином, з найменшим використанням ресурсів пам'яті та процесора

- використовувати ШІ заборонено під час виконання тестового завдання - нам важливі саме ваші думки та бачення (умови задач побудовані так, щоб рішення згенеровані за допомогою ШІ виділялись поміж інших)