Test Description

You are intelligence agents.
Your task is to receive data about people and phone call records, and use this data to find the **most dangerous suspects**.

You must send requests to the intelligence server to get:

- A list of people
- A list of call records

After that, you must analyze the data and find dangerous people by **connecting the call records with the people's data**.

Data Structure

Each data received from the server is a raw (**not json - text -** use res.text() not res.json()) array of objects - people or transcriptions.

All data is received from the server. You do **not** need to create functions to add, edit, or delete data.

People

Each person has:

- Name
- Age
- Profession
- Number of children
- Hobbies
- Favorite foods

Call Records/Transcriptions

Each record has:

- ID
- Call content – text that represents the recorded call
- Age of the person speaking

Note: IDs may appear more than once.

The keys may be spelled differently from how they are described here - double check it!

Required Menu + Actions

You must create a menu that supports the following actions:

---

1. Get People List

- Send a request to the intelligence server - general url + **“/people”**
- Receive an array of people objects.
- Save **only the array** into a file named `PEOPLE.json`.
- Do not save any extra text or metadata - keys, variable declarations etc. JUST THE ARRAY

---

2. Get Call Records/Transcriptions

- Send a request to the intelligence server - general url + **“/transcriptions”**
- Receive an array of call record objects.
- Save **only the array** into a file named `TRANSCRIPTIONS.json`.

---

3. Search People by Name

- Load the people list from the file. Use the fs module - callback or promises.
- Convert the text into JSON using `JSON.parse` (if needed)
- Ask the user to enter a name.
- Search for the person:
    - If found – print the person object.
    - If not found – print a message saying the person was not found.

---

4. Search People by Age

- Same steps as “Search by Name”.
- Instead of name, search by age.
- behind the scenes you can use the same function as before, if you want.

---

5. Find Dangerous People

Goal

Find which people send the most dangerous messages. We do that by first processing the transcriptions and finding which age has the highest average danger score (calculated by processing all the content messages) - and then finding the people with the same age.

Dangerous Calls

A call in the transcriptions data is considered **dangerous** if the content of this transcription includes any of these words:

- `death`
- `knife`
- `bomb`
- `attack`

Rules:

- Case does not matter (uppercase or lowercase).
- Each appearance of a dangerous word gives **1 point**, including duplicates.

Example:

content: i hate to cut carrots with my knife, i want death, here is my knife

Danger level: **3**
(2 times `knife`, 1 time `death`)

suggested steps:

Steps:

1. Go through **all call transcriptions**.
2. Calculate the danger level for each call's content.
3. Assign the danger level to the **age written in the call transcription**.
   For example: <age> : <array of danger levels, 1 per msg>
   {
   44: [12,34,4,1,3] // ,age 44 has 5 messages, with the scores 12…
   }
   you don't have to save values if there is no danger score (i.e. = 0)
4. Calculate the **average danger level per age**. Store this in a new object.
5. Find the **top 3 ages** with the highest average danger level.
6. From the people list, find **all people whose age is one of these top 3 ages**. Save it in a variable.
7. send the list as a url param to general url + **"/report"**. Print the response.

See appendix A at the end for some functions that might help!

Bonus Task – Step 1: Match Calls to People by Age

What You Are Trying to Do

You need to **connect call records to people**.

There is **no direct ID** that connects a call to a specific person.
The **only common field** between people and calls is **age**.

So, the rule is:

> A call belongs to a person **if the age in the call record equals the age of the person**.

Important Clarification

- This is **not** a one-to-one connection.
- This is **many-to-many**.

That means:

- One person can be linked to **many calls**.
- Many people can be linked to **the same calls** if they share the same age.
- A person can have **zero calls**.

This is expected and correct.

Step 2 – Personal Danger Score

For each person, calculate a **personal danger score** using the rules below:

1. **Call Danger Score**
    - Sum the danger level of all calls linked to the person's age.
2. **Profession Risk Bonus**
    - If the profession contains any of the following words, add **+2 points**:
        - `security`
        - `military`
        - `chemical`
        - `engineering`
    - Case does not matter.
3. **Hobby Risk Bonus**
    - If the hobbies list contains any of the following, add **+1 point per match**:
        - `shooting`
        - `martial arts`
        - `explosives`
        - `hunting`
4. **Family Size Adjustment**
    - If number of children is **0**, add **+1 point**.
    - If number of children is **3 or more**, subtract **1 point**.

Appendix A - some functions that might help:

Object → Array of [key, value]

```
const obj = {
  name: "Dana",
  age: 28,
  city: "Haifa"
};
const result = Object.entries(obj);
```

Result

```
[
  ["name", "Dana"],
  ["age", 28],
  ["city", "Haifa"]
]
```

Object → Array and Sort by Value

```
const scores = { Tom: 12, Lea: 30, Ron: 20 };
const sorted = Object.entries(scores)
  .sort((a, b) => a[1] - b[1]); // reverse if you want from big to small
```

Result:

```
[["Tom", 12], ["Ron", 20], ["Lea", 30]]
```

General grading:

Get People List	Request + save raw array to PEOPLE.json, menu option	15
Get Call Records	Request + save raw array to Transcriptions.json, menu option	15
Search by Name	Correct search logic, data load, menu option	12
Search by Age	Correct search logic, data load, menu option	12
Dangerous Call Detection	Detect words, case-insensitive, count duplicates	15
Dangerous Age Analysis	Avg danger per age, top 3 ages, select people	20
Final Output + menu	Show correct list of dangerous people, general menu option for dangerous search	11
Total (Main)		**100**