Exercise 1

You are given two strings `s` and `t`, representing biological sequences (e.g., DNA or protein strings).

The **edit distance** between `s` and `t` is defined as the minimum number of single-character operations required to transform `s` into `t`.
The allowed operations are:

- insertion of a character,
- deletion of a character,
- substitution of one character with another.

Each operation has a cost equal to 1.

Example

Input

```
s = "PLEASANTLY"
```

```
t = "MEANLY"
```

Expected output

```
5
```

(The minimum number of edit operations needed to transform `s` into `t` is 5.)

What to deliver

1. **Pseudocode** for an algorithm that computes the edit distance between two strings `s` and `t` using **dynamic programming**.
    - Clearly define the meaning of each entry in the DP table.
    - Specify the base cases.
    - Specify the recurrence used to fill the table.
    - Indicate what value is returned as the final result.
2. The **time complexity** and **space complexity** of your algorithm, expressed using Big-O notation.

Exercise 2

You are given a list of dictionaries, each representing the expression level of a gene measured in a biological sample. Each dictionary contains the following fields:

- **sample_id**: unique identifier of the sample
- **expression**: measured gene expression level
- **condition**: either `"treated"` or `"control"`

A researcher wants to evaluate the average gene expression **only in treated samples**.

Your task is to use the **MapReduce paradigm** (i.e., **map**, **filter**, and **reduce**) to:

1. Compute the average expression level of the samples belonging to the **treated** condition.
2. Return the result as a dictionary only if the average expression is **strictly greater than 50**. Otherwise, return an empty list.

Example input

```
samples = [

          {"sample_id": 1, "expression": 62, "condition": "treated"},

          {"sample_id": 2, "expression": 35, "condition": "control"},

          {"sample_id": 3, "expression": 58, "condition": "treated"},

          {"sample_id": 4, "expression": 41, "condition": "control"},

          {"sample_id": 5, "expression": 55, "condition": "treated"},

          {"sample_id": 6, "expression": 48, "condition": "treated"},

          {"sample_id": 7, "expression": 67, "condition": "treated"}

]
```

Expected output

```
[{"average_expression": 58.0} ]
```

Notes

- Use **filter** to select only samples belonging to the `"treated"` condition.
- Use **map** to extract the expression values.
- Use **reduce** to compute the sum of the expression values.
- A solution that does **not** follow the MapReduce paradigm is **not valid**.

Exercise 3 — Packing Biological Samples

A laboratory has collected **n** biological samples that must be transported to another facility. Each transport box can contain **at most k grams** of material.

Each sample has a given weight, and **samples cannot be split**. A box may contain multiple samples as long as the total weight does not exceed **k**.

To simplify the packing process, the laboratory decides to process the samples, organizing the samples with some logic and then filling each box as much as possible before opening a new one.

The goal is to determine **the minimum number of boxes** required using this strategy.

Input format

- An integer **k** representing the maximum capacity of each box.
- A list **weights** of **n** integers, where `weights[i]` is the weight (in grams) of the *i*-th sample.

Example

Input

```
weights = [3, 8, 2, 7, 5, 6]

k = 10
```

Expected output

```
4
```

One possible packing produced by the strategy is:

```
Box 1: 8 + 2

Box 2: 7

Box 3: 6

Box 4: 5 + 3
```

What to deliver

1. Write pseudocode for an algorithm that computes the minimum number of boxes required.
2. Explain which type of algorithm you are using (e.g., greedy, dynamic programming, etc.) and justify why it is appropriate for this problem.
3. Give the time complexity and space complexity of your solution in Big-O notation.