LIST - TRAINING TASKS – LAMBDA

- I will only accept tasks submitted **by the deadline**.

- I'll give you feedback on whether the solution is correct or not. The most important thing to me in your code is the **output**.

- The program **must display** some information (e.g., popups or messages) when it receives input or is running in a loop, indicating what action the user should take.

- It will **not** be possible to resend the code.

- An incorrect solution may negatively affect your activity grade.

Task 1: Passing Multiple Arguments and Returning Complex Data

Objective: Work with multiple arguments, return complex data structures, and apply advanced list operations.

Description: Write a Python program that:

1. Defines a function `process_data(data)` that takes a list of tuples, where each tuple contains two elements: a string and a number.
2. The function processes the data and:
   - Sorts the tuples by the string (ascending).
   - Filters out tuples where the number is less than 5.
   - Maps the numbers to their squares.
3. The function should return a dictionary with:
   - A sorted list of strings.
   - A list of squared numbers.
   - A count of how many tuples were filtered out due to the number being less than 5.

Instructions:

- Use multiple arguments (tuples of data) and return complex results in the form of a dictionary.
- Apply advanced list operations such as sorting, filtering, and mapping.

```
data = [("apple", 3), ("banana", 7), ("cherry", 2), ("date", 10)]
```

Task 2: Advanced Data Processing with Multiple Layers of Sorting, Filtering, and Transformation

Objective: Work with multiple arguments, apply multiple levels of sorting, more complex filtering, and transformation using built-in functions.

Description:

Write a Python program that:

1. Defines a function `process_data(data)` that takes a list of tuples, where each tuple contains three elements:
   - A string (name).
   - A number (age).
   - A boolean value (status, where `True` means the person is active, and `False` means inactive).
2. The function processes the data and:
   - **Sorts** the tuples in descending order based on the age.
   - Filters out tuples where the number (age) is less than a given threshold (e.g., 18), or where the status is `False`.
   - Maps the names to uppercase.
   - Calculates the average age of the filtered data.
3. The function should return a dictionary with:
   - A sorted list of names (in uppercase).
   - A list of ages for the filtered data.
   - The average age of the filtered data.
   - The count of active people.

Instructions:

- Use multiple arguments (tuples with data of type string, integer, and boolean) and return complex results in the form of a dictionary.
- Apply advanced operations such as sorting by multiple criteria (age descending, status), filtering with conditions based on age and status, and transforming data (mapping names to uppercase).
- Handle cases where there are no people who meet the filtering conditions by calculating the average age appropriately (return `None` if no valid entries).

```
data = [ ("Alice", 30, True),  ("Bob", 15, True),  ("Charlie", 25,
False), ("David", 40, True), ("Eve", 17, False), ("Frank", 20, True)]
```