

## CS103 – Fall 2025 – Lab12

### General Instructions

1. In today's lab, you are required to complete all required exercises during class to receive attendance credit.
2. You are welcome to:
  - o Work together with classmates.
  - o Search online for help or documentation.
  - o Ask the TA for guidance if you are stuck.
3. To receive credit, you must finish and show your solutions to the TA before leaving lab.
4. There are also extra (optional) exercises at the end for those who want to challenge themselves. These are not required for attendance credit but are recommended for practice.

### Exercise Instructions

- Make a folder **Lab12** inside your **cs103fa25** folder.
- Create a new python file inside your Lab12 folder (**lab12.py**).

## Required Exercises

### EXERCISE 1:

Write a function “**hailstone**” that takes an int “**n2**” and prints the hailstone sequence.

**Hailstone Numbers:** This sequence takes the name hailstone as the numbers generated bounce up and down. To generate the sequence, follow these rules:

- a. If the number is odd multiply it by 3 and add 1
- b. If the number is even, divide by two.
- c. Repeat until you reach number 1.

### Sample Inputs & Outputs

**n2 = 3** -> 10, 5, 16, 8, 4, 2, 1

**n2 = 6** -> 3, 10, 5, 16, 8, 4, 2, 1

**n2 = 7** -> 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

**EXERCISE 2:**

Write a function “**keysToRemove**” that takes a dictionary “**d1**” and a list “**l1**” and returns the modified dictionary. The function will read each element of the list and remove them from the dictionary. Assume that the list elements are the equal to the dictionary keys.

**Sample Input 1:**

```
d1 = {"FirstName": "Mark", "LastName": "Zuckerberg", "Salary": 2000000,  
"City": "Palo Alto", "Company": "Facebook"}  
  
l1 = ["Salary", "Company"]
```

**Sample Output 1:**

```
{"FirstName": "Mark", "LastName": "Zuckerberg", "City": "Palo Alto"}
```

**EXERCISE 3:**

Write a function “**animalLegs**” that takes a dictionary “**d2**” and returns an integer. The function will read each key/value pairs from the dictionary and calculate the total number of the legs of the given animals. Assume that the dictionary only includes the following animals:

chickens = 2 legs, cows=4 legs, rabbits=4 legs, dogs=4 legs

**Sample Input 1:**

```
d1 = {"chickens":10, "rabbits":5}
```

**Sample Output 1:**

40

Hints:  $2*10 + 4*5 = 40$

**Sample Input 2:**

```
d2 = {"dogs":10, "chickens":5, "cows":5}
```

**Sample Output 2:**

70

Hints:  $4*10 + 2*5 + 4*5 = 70$

## **Extra (Optional) Challenges**

These are not required for credit but will help you practice and strengthen your problem-solving skills.

### **Challenge 1: "listDetails"**

Write a function **listDetails** that takes in a list **L** and returns a tuple containing (in the following order) the *number of elements in the list*, the *minimum value*, the *minimum value's index*, the *mean* (rounded to the nearest hundredth), the *maximum value* and the *maximum value's index* (total of **six** elements). Assume that the input will always be a list (no assertions are necessary). Hint: You have already written code this semester for most of this problem. Feel free to use helper functions, if needed, to accomplish this task. Built-in-methods and functions are permitted.

#### **Sample input:**

**L** = [-8, -23, 18, 103, 0, 1, -4, 631, 3, -41, 5]

#### **Sample Output:**

(11, -41, 9, 62.27, 631, 7)

### **Challenge 2 : minSum**

Write a function "**minSum**" that takes a list "**L2**" which includes integer numbers. This function returns another list that includes the indices of the two smallest elements such that they add up to the smallest sum in the input list. You may assume that each input would have exactly one solution, and you may not use the same element twice. Our goal here is to find the minimum sum of any two numbers.

#### **Sample Input:**

**L2** = [2, 78, 9, 21, 4, 99]

#### **Sample Output:**

[0, 4]

Hints: minimum sum is 2+4=6

**Challenge 3: Word Ladder Tracker**

Write a function `wordLadder(sentences)` that takes a list of sentences and returns a dictionary where:

- Keys are **words** that appear in the sentences.
- Values are **lists of indices** indicating which sentences the word appeared in.

**Hints:**

- Words should be case-insensitive (convert all words to lowercase).
- You can split each sentence into words using `.split()`.
- Use a dictionary to map words to lists of sentence indices.

**Sample Input:**

```
sentences = ["Hello world", "Hello Python world", "Python is fun"]
```

**Expected Output:**

```
{'hello': [0, 1], 'world': [0, 1], 'python': [1, 2], 'is': [2], 'fun': [2]}
```

**Challenge 4: Nested Inventory Tracker**

Write a function `inventorySummary(items)` that takes a list of dictionaries representing inventory transactions and returns a **summary dictionary** showing total quantities of each item.

- Each dictionary in the list has the format: `{"item": "apple", "quantity": 5}`.
- If the same item appears multiple times, sum up the quantities.

**Hints:**

- Use a dictionary to accumulate the counts.
- You can use `dict.get(key, default)` to simplify updating totals.

**Sample Input:**

```
items = [
    {"item": "apple", "quantity": 5},
    {"item": "banana", "quantity": 2},
    {"item": "apple", "quantity": 3},
    {"item": "orange", "quantity": 4},
    {"item": "banana", "quantity": 1}
]
```

**Expected Output:**

```
{'apple': 8, 'banana': 3, 'orange': 4}
```

If you finish early, try the optional challenges!