CS103 – Fall 2025 – Lab10

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:
   - Work together with classmates.
   - Search online for help or documentation.
   - 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 **Lab10** inside your **cs103fa25** folder.
- Create a new python file inside your Lab10 folder (lab10.py).

Required Exercises

EXERCISE 1:

Write a function "**isSorted**" that takes a list "**L**"**.** The function will print "`Sorted`" if the list is sorted, otherwise it will print "`Unsorted`". The list can be sorted by increasing or decreasing order.

Sample Input

```
L = [26,8,6,2,1]
```

Sample Output

```
Sorted
```

EXERCISE 2:

Write a function “**`factorial`**” that takes an int “**`n`**” and returns n! value. ***Use Recursion to solve this exercise.***

Sample Input:

```
n = 5
```

Sample Output:

```
120
```

EXERCISE 3:

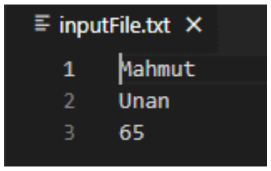
Write a function “**`fChecker`**” that doesn’t take any input. The function reads “`inputFile.txt`” and reads it line by line. Assume the first line is the first name, second line is the last name, and the third line is the age of the person. *You will have to create this inputFile.txt.* The function should print the greetings message with the expected age in 5 years. **Hint:** To read the file line by line, you can use readline() function:

```
fo = open("foo.txt", "r") # Open a file

line = fo.readline()# Read the file line by line
```

Also, the output variable (line) will include an `endofline (\n)` character. You will need to remove it.

Sample Input



```
Mahmut
Unan
65
```

Sample Output

```
Hello, Mahmut Unan. You will be 70 years old in 5 years.
```

Extra (Optional) Challenges

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

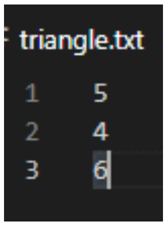
Challenge 1: "triangleChecker"

Write a function "**triangleChecker**" that doesn't take any input. The function reads "triangle.txt" which includes three numbers (each line has one number). *You will have to create this text file.* Assume these values are the side lengths of the triangle. The function should check the validity of the triangle and print whether the triangle is valid or not. If it is valid, the result should also indicate the type of triangle (equilateral, isosceles, or scalene).

Here are some hints about triangles:

- Triangle inequality only depends on the length of the sides.
- An **equilateral** triangle is a triangle in which all three sides are equal.
- A **scalene** triangle is a triangle that has three unequal sides.
- An **isosceles** triangle is a triangle with (at least) two equal sides

Sample Input



```
triangle.txt
5
4
6
```

Sample Output

```
The triangle is valid.
It is a scalene triangle.
```

Challenge 2: Recursive Sum of Nested List

Write a function **recursiveSum(L)** that takes a **nested list** of integers and returns the **sum of all numbers**, no matter how deeply nested the lists are.

Sample Input

```
L = [1, [2, [3, 4], 5], 6]
```

Sample Output

```
21
```

Challenge 3: `countWords(sentences)`

Write a function **`countWords(sentences)`** that takes a list of sentences (strings) and returns a **dictionary** where the keys are words and the values are **counts of how many times each word appears** across all sentences.

Sample Input

```
sentences = ["hello world", "hello python", "python is fun", "hello
fun"]
```

Sample Output

```
{'hello': 3, 'world': 1, 'python': 2, 'is': 1, 'fun': 2}
```

Challenge 4: Boolean Path in Nested Grid

Write a function **`hasPath(grid)`** that takes a 2D list (nested list) of **boolean values** representing a grid (`True` = open, `False` = blocked).
Return `True` if there is **any row** where **all values are `True`**, and `False` otherwise.

Sample Input:

```
grid = [

    [False, True, True],

    [True, True, True],

    [False, False, True] ]
```

Sample Output

```
True
```

To get attendance credit, finish Exercises 1–3.

If you finish early, try the optional challenges!