CS 112 **Fall 2022**

Lab 8 : Recursion

Assignment is due by the end of the lab and must be submitted through Blackboard.

Submission instructions : you must submit one (1) Python file. Please name the file in the usual *NetID_2XX_Lab8.py* format.

Problem Statement :

You are provided a recursive function which takes in a non-negative integer $n \geq 0$ and calculates the corresponding Fibonacci number. You are also provided a helper function which takes in any 2-dimensional list as input and prints it in a nice format and provided a quick test case tester code. *Simply download the Lab8Template.py file from Blackboard under the "Lab Slides" link to obtain the provided code.*

Brief explanation of the provided code:

```
def fibonacci(n):
    """
    This is a recursive function provided to you. Given the value of n,
    it will return the nth Fibonacci number.
    """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

def print_matrix(arr):
    """
    This is a helper function provided to you. You can use this to print a
    2D list (matrix) in a nice format.
    """
    for row in arr:
        for item in row:
            print(item, end="\t")
        print()
```

Your Task:

Assuming i and j are index locations of arr[i][j], define function fill_matrix(arr) such that you fill the 2D array arr that is passed in with the following guidelines:

- If *i* and *j* are both zero, assign "-" to that location
- If *i* is greater than *j*, then assign to that location the result of the division of the i^{th} Fibonacci by the j^{th} Fibonacci (*hint:* use the helper function provided)
- If *i* is less than *j*, then assign to that location the result of the division of the j^{th} Fibonacci by the i^{th} Fibonacci (*hint:* use the helper function provided)
- If *i* is equal to *j*, assign to that location a string value "*"

The function signature along with all the above "to do" items is listed below:

```
def fill_matrix(arr):
    """
    TODO (Assignment) : Complete this function.

    Given a 2D array (matrix) of any size, fill it using
    Fibonacci numbers such that :

    if i > j, arr[i][j] = fibonacci(i) / fibonacci(j)
    if i < j, arr[i][j] = fibonacci(j) / fibonacci(i)
    if i == j, arr[i][j] = "*"

    if i == 0 or j == 0, arr[i][j] = "-"
    """
    return arr
```

To do the assignment, use the *Lab8Template.py* file. Then you can run the file using the interpreter:

```
python3  <solution_filename.py>
```

Once you have completed the function, simply run the file and you should get something like the image below, which shows the output expected for a 7 × 7 matrix.

```
Test case passed!
Output :

-       -       -       -       -       -       -
-       *       1.0     2.0     3.0     5.0     8.0
-       1.0     *       2.0     3.0     5.0     8.0
-       2.0     2.0     *       1.5     2.5     4.0
-       3.0     3.0     1.5     *       1.7     2.7
-       5.0     5.0     2.5     1.7     *       1.6
-       8.0     8.0     4.0     2.7     1.6     *
```

In case your output does not match the expected output, you will get a message that shows the expected output and your output one after another.

Note: the sample test case is only provided for a 7 × 7 matrix. If you want to test your code with some different sized matrix, first ensure that you comment out the part of the code which compares the output array with an expected array in the template file provided.

Important Guidelines :

- Use the recursive function definition provided to complete the *fill_matrix* function.
- Use math/relational/logic operators such as */*, <, >, ==, *or,* etc. as needed.
- Use *if else* statements as needed.
- Use 2D lists as needed.
- You should not use any built-in functions other than *range()* and *len().*
- You should **not** use any string-formatting or print or input statements anywhere in your code.
- You may need to use casting to *float* and the *round* function to match the expected output.
- You should **not** import any other modules (for example, math).
- Do **not** use any other programming element that has not been covered in the class or the Zybook readings at this time.
