Lab Week 15 Grading Rubric and Instructions

This lab is assigned for Week 15 of COM S 127: Introduction to Programming.

This lab is due by the end of the lab period seven (7) days after the one it is assigned in. See the syllabus for details.

Lab Objective

The purpose of this lab is to give students exposure to custom-made classes and basic object-oriented programming techniques by way of the Runestone readings. It will also give students practice with problems similar to those that will be found on the final exam.

Instructions/ Deliverables

NOTE: These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

Reading:

- Read the following articles, and prepare written summaries of each topic in your Engineering Notebook to show the TA.
  - Subset sum problem - by: Wikipedia
    - Available: https://en.wikipedia.org/wiki/Subset_sum_problem
  - How to Solve Two Sum - by: Tom Wagner and Dominic Platt
    - Available: https://interviewing.io/questions/two-sum
  - Fizz buzz - by: Wikipedia
    - Available: https://en.wikipedia.org/wiki/Fizz_buzz#:~:text=Fizz%20buzz%20is%20a%20group,with%20the%20word%20%22fizzbuzz%22
  - Fizz Buzz Problem - by: enjoy algorithms
    - Available: https://www.enjoyalgorithms.com/blog/fizz-buzz-problem
- Read the following sections from the online Runestone textbook and show the TA the notes you took in your Engineering Notebook for each chapter once you are done.
  - Chapters 17, 18, and 19.
    - **NOTE**: You do not need to complete any of the exercises at the end of the chapter. However, it would be helpful to you in the long term if you were to do so.
    - Available: https://runestone.academy/ns/books/published/thinkcspy/index.html?mode=browsing

NOTE: The following programming tasks comprise a list of potential programming questions for the final exam. At least one (1) if not more of these questions will appear on the final exam. Take this opportunity to figure them all out as a way of studying for the test.

NOTE: If you "borrow" any of the code found in the readings above to complete the tasks below, or if you take code from *any* source that is not original to you, you **must** provide proper citation/ attribution. This includes the author's name, the date the article was written (if available), when you accessed the article, and the URL where the article can be found. Place this citation in comments for the function that the code appears in.

`twoSum.py`

- Create a new Python script.
  - Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
  - In `main()`, hardcode a list of integers and a target value.
  - Pass the list of integers and target value to a function called `twoSumLoops()`, which will solve the two-sum problem in the naive way shown in the article.
    - This function will then return the pair of indices back to `main()`, where it should be printed out.
  - Pass the list of integers and target value to a function called `twoSumDict()`, which will solve the two-sum problem in the more efficient way shown in the article.
    - This function will then return the pair of indices back to `main()`, where it should be printed out.
  - Create two new functions, called `twoSumLoopsAll()` and `twoSumDictAll()`, which find and return *all* possible index pairs in the integer array - not just one. Return all of these pairs as a list of lists.
  - Your final submission will have a total of four (4) functions in it, all of which run on the same input.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `twoSum.py`.
- **NOTE**: If you quote/ use/ reference any code from a reading/ article/ the internet, you **must** provide proper citation/ attribution.

`fizzBuzz.py`

- Create a new Python script.
  - Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
  - In `main()`, take in an integer as user input.
  - Pass the integer you took as input to a function called `fizzBuzzModulus()`, which will solve the 'fizz buzz' problem using the modulus operator approach, as seen in the article.
    - Add the feature of printing 'Bazz' when the number is divisible by seven (7).
    - Return your output as a list to `main()` and print it out.
  - Pass the integer you took as input to a function called `fizzBuzzDict()`, which will solve the 'fizz buzz' problem using the dictionary (hash table) approach, as seen in the article.
    - Add the feature of printing 'Bazz' when the number is divisible by seven (7).
    - Return your output as a list to `main()` and print it out.
  - Your final submission will have a total of two (2) functions in it, all of which run on the same input.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `fizzBuzz.py`.
- **NOTE**: If you quote/ use/ reference any code from a reading/ article/ the internet, you **must** provide proper citation/ attribution.

`isPalindrome.py`

- Create a new Python script.
  - Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
  - In `main()`, take in a string as user input.

    - Pass the string you took as input to a function called `isPalindromeIterative()`, which will check to see if the string is a palindrome in an iterative manner.
        - This function will then return the True/ False value of the function back to `main()`, where it should be printed out.
    - Pass the string you took as input to a function called `isPalindromeRecursive()`, which will check to see if the string is a palindrome in a recursive manner.
        - This function will then return the True/ False value of the function back to `main()`, where it should be printed out.
        - **NOTE**: Do *not* attempt to use a loop when completing this task - only do it recursively.
    - Your final submission will have a total of two (2) functions in it, all of which run on the same input.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `isPalindrome.py`.
- **NOTE**: If you quote/ use/ reference any code from a reading/ article/ the internet, you **must** provide proper citation/ attribution.

`reverseString.py`

- Create a new Python script.
    - Use a `main()` function in the way demonstrated in class.
        - Ex: `if __name__ == "__main__":` etc.
    - In `main()`, take in a string as user input.
    - Pass the string you took as input to a function called `reverseIterative()`, which will reverse the string in an iterative manner. (Do *not* use the string slicing method for this.)
        - This function will then return the reversed string back to `main()`, where it should be printed out.
    - Pass the string you took as input to a function called `reverseRecursive()`, which will reverse the string in a recursive manner.
        - This function will then return the reversed string back to `main()`, where it should be printed out.
        - **NOTE**: Do *not* attempt to use a loop when completing this task - only do it recursively.
    - Your final submission will have a total of two (2) functions in it, all of which run on the same input.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `reverseString.py`.
- **NOTE**: If you quote/ use/ reference any code from a reading/ article/ the internet, you **must** provide proper citation/ attribution.

Stack Drawings

- Create two drawings, either by hand or with the a drawing program in the computer, where you trace the 'call stack' of **two other exercises in this lab**. These drawings should appear in your engineering notebook.
    - Each time a function is called, add a new step to the drawing with a single box - i.e. *push* the function onto the stack. This box should be labeled with the function name, and include the current values of its parameters. If there are no parameters, just include empty parentheses, as you would with a normal call to such a function. e.g. `main()`
    - Each time a function returns, add a new step to the drawing where the box that corresponded to the function that just returned is removed - i.e. *pop* the function off of the

stack.
        - Each time the 'call stack' changes, you will re-draw the previous step in a new location, and either add or remove the appropriate box.
    - For the `print()` function (which would, indeed, be pushed to/ popped off the stack), just write `print()`, and what the `print()` function would print to the terminal inside the parentheses. e.g. `print("tac")` or `print(True)`.
    - Do not include functions such as `range()`, `len()`, `int()`, `str()`, etc. in your drawing.
        - While these are all, indeed, functions, and would, indeed be pushed to/ popped off the stack, the main point of this exercise is to see how that code that *you* have written interacts with the runtime-stack.
    - **NOTE**: This drawing will depict a series of 'steps' of adding and removing function calls to and from the stack. It could end up being a drawing that is quite large, with many different 'steps' depicted. **This is NOT a trivial task**.
- **HINT**: You can use the VS Code debugger to trace the 'call stack' and the function variables.
- **HINT**: See the lecture slides about 'exceptions' for an example of what each step in this 'call stack' drawing might do.
- **HINT**: Have a *very* firm idea of how the stack works **before** you start your drawings. They may be very large (perhaps multiple pages), and you do not want to have to re-do them if you make a mistake.

Optional Readings

NOTE: These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

Ways to avoid social engineering attacks - by: usa.kaspersky.com

- Available: https://usa.kaspersky.com/resource-center/threats/how-to-avoid-social-engineering-attacks

How to Get a Job with a Computer Science Degree and No Experience - by: genspark.net

- Available: https://genspark.net/how-to-get-a-job-with-a-computer-science-degree-and-no-experience/
- **NOTE**: I do not advise having *no* experience when you graduate. Try your best to get at least one or two internships while you are in school. The time to start working on this issue is *right now* [MH].

How to network: 17 tips for shy people - by: cio.com

- Available: https://www.cio.com/article/230572/how-to-network-17-tips-for-shy-people.html

Why Do Game Developers Suck So Much? - by: Mat Growcott

- Available: https://www.gamesreviews.com/articles/03/why-do-game-developers-suck-so-much/

Files Provided

None

Example Script

reverseString.py

```
# Matthew Holman                4-24-2023
# Lab Week 15 - An example script layout

def reverseIterative(string):
    pass

def reverseRecursive(string):
    pass

def main():
    pass

if __name__ == "__main__":
    main()
```

Example Output

Running: `python .\reverseString.py`

```
Input a String: cat
Iterative: tac
Recursive: tac
```

Grading Items

- (**Attendance**) Did the student attend the lab meeting, or make arrangements to attend virtually via WebEx?: _____ / 1
- (**Reading**) Has the student read the assigned articles, as well as chapters 17, 18, and 19 of the Runestone textbook and shown their notes in their Engineering Notebook to the TA?: _____ / 3
- (**`twoSum.py`**) Has the student completed the task above, and saved their work to a file called `twoSum.py`?: _____ / 1
- (**`fizzBuzz.py`**) Has the student completed the task above, and saved their work to a file called `fizzBuzz.py`?: _____ / 1
- (**`isPalindrome.py`**) Has the student completed the task above, and saved their work to a file called `isPalindrome.py`?: _____ / 1
- (**`reverseString.py`**) Has the student completed the task above, and saved their work to a file called `reverseString.py`?: _____ / 1
- (**Stack Drawings**) Has the student completed the drawing task above - either by hand or with the computer?: _____ / 2

TOTAL _____ / 10