Lab #11 Grading Rubric and Instructions

This is the eleventh lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

Lab Objective

The purpose of this lab is to help you get started working with `git`, and to show you some of the things you will need to do to pass a technical interview for an internship or a job after graduation.

Instructions/ Deliverables

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

CITATION: Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
    - ex: [MH]

Lecture Notes/ Supplemental Readings:

- 'Check off' your notes in your Engineering Notebook for the following material with the TA/ Instructor. These notes should already be done before the start of the lab period.
    - Runestone chapter 16
        - **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.
        - **NOTE**: You only need to take notes on the Runestone readings if you did *not* take lecture notes in class. The contents of these chapters are what was covered in the lectures.
    - `23_Recursion.pptx`
        - **Available**: `'Files' -> 'Supplimental_Slides_and_Files'` folder on Canvas
    - Facts and myths about Python names and values - by: Ned Batchelder
        - **Available**: https://nedbatchelder.com/text/names.html
    - Why is FAANG so obsessed with Whiteboard Coding Interviews? - by: Fabian Hinsenkamp
        - **Available**: https://medium.com/codex/why-is-faang-obsessed-with-whiteboard-coding-interviews-21a1ae2b693a
    - 5 Whiteboard Coding Tips for Interviews - by: John Nagro
        - **Available**: https://product.hubspot.com/blog/5-whiteboard-coding-tips-for-interviews
    - Tips to use GitHub as your Portfolio - by: Pachi Parra
        - **Available**: https://www.linkedin.com/pulse/tips-use-github-your-portfolio-pachi--yw4vf
    - How to Solve Two Sum - by: Tom Wagner and Dominic Platt
        - **Available**: https://interviewing.io/questions/two-sum
    - Fizz Buzz Problem - by: enjoy algorithms
        - **Available**: https://www.enjoyalgorithms.com/blog/fizz-buzz-problem

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.

GitHub: [MH]

- **WARNING - WARNING - WARNING**: It is entirely possible to accidentally delete an entire semester's worth of work when doing this task.
    - If you attempt to do any sort of 'discard changes' operation to your initial files *before* you have done an initial commit, or some sort of `git rm -rf` operation that you are unsure of, you could accidentally delete all of your files.
    - **MAKE EXTRA SURE THAT YOU MAKE ONE OR TWO EMERGENCY BACKUPS OF YOUR WORK IN DIFFERENT FOLDERS ON YOUR COMPUTER BEFORE ATTEMPTING THIS TASK**
    - **IN FACT, YOU SHOULD MAKE A COPY OF ALL YOUR WORK, PUT EVERYTHING INTO A ZIP FILE, AND EMAIL IT TO YOURSELF - JUST IN CASE!**
    - **YOU HAVE BEEN WARNED!**
- Complete the following tutorial: https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners
    - Be sure to install `git` on your computer.
        - If you are on a mac, you should already have it. You can check on the terminal with the command `git --version`.
    - Be sure to create a free account on GitHub:
        - https://github.com/
    - If you are using a mac, consider the information in the following article which details how to remove pesky `.DS_Store` files:
        - https://www.geeksforgeeks.org/how-to-remove-ds_store-files-from-git-repositories/
    - Do the tutorial with some 'dummy files' for safety before trying it with your real work.
    - Please note that the `touch` command seen in the tutorial not available on Windows CMD or PowerShell - it is a Mac/ Linux/ Unix command.
- After you have an idea of how `git` works, make a new public repository for all of the code you created for this class.
    - Give this repo a name like "Iowa State University - COM S 1270 Code" or something like that.
    - Be sure to organize your code into folders such as `assignments`, `labs`, etc.
    - The repository you 'check off' with the TA/ Instructor should have all of your code for the class committed and pushed.
- Create a second public repository, specifically for storing your 'Whiteboard' programming solutions.
    - Give your repo a name like "Whiteboard Problem Solutions" or something like that.
    - The repository you 'check off' with the TA/ Instructor should have your solutions for the 'NeetCode' problems listed in the `NeetCode` task below, as well as the other questions which are listed below as well.
- Once you have created and populated both of your repos, show them to the TA/ Instructor to 'check them off.'

NeetCode: [MH]

- LeetCode is an online repository of programming questions, many of which are similar to the various test questions and programming challenges we have been doing all semester. These types of questions are a staple of 'whiteboard technical interviews,' and are thus something you should start spending more time working on in preparation for your inevitable search for a job.
- NeetCode is an interesting resource in that it will guide you on a path to being able to solve LeetCode problems.
    - There is not only an ordering to the questions for you to answer, but there are often videos that explain the various solutions to the problems.
- Create a free account on https://neetcode.io/ and explore the "Roadmap" a bit - starting at the top and going down the tree.
    - Please note - you do not have to subscribe or buy anything.
- Once you have explored the site a bit, click on the "Arrays & Hashing" node of the tree.
- Attempt each of the three problems below:
    - "Contains Duplicate"
    - "Valid Anagram"
    - "Top K Frequent Elements"
- After trying your best to solve each of the above questions, watch and take notes on the solution videos for each.
    - You should then have working solutions for each of these questions.
    - You can use the online code editor to test your code, but you should also save your code locally to a file you create in VS Code. This will allow you to put your solution in your `git` repo later.
        - Be sure to use the usual `if __name__ == "__main__":` setup, as well as several 'test cases' for each problem.
- Once you have successfully solved all three problems above, show your local VS Code files to the TA/ Instructor to 'check off' this task.

`twoSum.py`: [MH]

- 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`: [MH]

- 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`: [MH]

- 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`: [MH]

- 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.

Attendance:

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional Resources for Study' found in the Canvas announcement of the same name.
    - **NOTE**: If you leave early, you will not receive the 'attendance points' for the lab.

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

The insider's guide to recursion interview questions - by: Educative

- Available: https://www.educative.io/blog/the-insiders-guide-to-recursion

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(s):
    pass

def reverseRecursive(s):
    pass

def main():
    testVal = input("Please Enter a String: ")

        answer1 = reverseIterative(testVal)
        print("Iterative:", answer1)

        answer2 = reverseRecursive(testVal)
        print("Recursive:", answer2)

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

Example Output

Running: `python reverseString.py`

```
Please Enter a String: cat
Iterative: tac
Recursive: tac
```

Grading Items

- (**Lecture Notes/ Supplemental Readings**) Has the student taken notes on the listed material, and shown their notes in their Engineering Notebook to the TA/ Instructor?: ______ / 20
- (**GitHub**) Has the student completed the task above, and shown their two populated repositories to the TA/ Instructor?: ______ / 20
- (**NeetCode**) Has the student completed the task above, and shown their solutions to the TA/ Instructor?: ______ / 10
- (**twoSum.py**) Has the student completed the task above, and saved their work to a file called twoSum.py?: ______ / 10
- (**fizzBuzz.py**) Has the student completed the task above, and saved their work to a file called

fizzBuzz.py?: ______ / 10

- (**isPalindrome.py**) Has the student completed the task above, and saved their work to a file called isPalindrome.py?: ______ / 10
- (**reverseString.py**) Has the student completed the task above, and saved their work to a file called reverseString.py?: ______ / 10
- (**Attendance**) Did the student attend the full lab meeting in person?: ______ / 10

TOTAL ______ / 100