Honors Computer Science Python
Mr. Clausen
Program 6A, 6B, 6C, 6D

PROGRAM 6A Rock Paper Scissors Lizard Spock (40 points)

Write a program to play the game Rock, Paper, Scissors, Lizard, Spock (RPSLS). The user will be playing against the computer. Ask the user for their choice, generate a random number as the choice for the computer, determine if the computer wins, if the player wins or if there is a tie, and keep a running total of who wins each round, the number of ties, and display the results each time the user chooses to play the game. This program can be written using if, elif statements to determine the winner, to convert from the user's choice to a corresponding string and to convert the computer's numeric choice to a corresponding string representing the computer's choice. You could use your own algorithm or the class could ask me to discuss an algorithm using the modulus operator.

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) import random

4) Leave a blank line after the import statement.

5) Type a one line comment of all equal signs.
   #==========

6) Define the **main** function. The main function should declare and initialize all of the variables (except any variables that are better left as local variables within other functions) and call other functions. The last line of code in the main function should be:
   input("Press enter to quit the program")

7) Initialize all of the variables that are to be used in this program **in the main function** and call all of the other functions from the main function. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

8) After the main function, type a one line comment of subtraction signs. **Include this comment after each function as well.**
   #----------

9) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function and run your program to see if it works without errors. **See step #18.**

10) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does, a "type contract", and some sample function calls with sample expected outputs.**

11) Define the function "menu" that displays the menu (as pictured below) and asks the user to enter their choice and returns this choice to the main function. Upper and lower case values should work for the menu choice to quit. Pass the variable representing the user's choice to this function when calling it from the main function.

```
Rock Paper Scissors Lizard Spock Main Menu
1. Rock
2. Spock
3. Paper
4. Lizard
5. Scissors
Q. Quit the game
```

12) In the main() function, create a "while loop" that will quit when the user enters either a lowercase 'q' or upper case 'Q' and keep running otherwise.
Now call the "menu" function from the main function **inside the "while loop"** and run your program to see if it works without errors, and will quit when the user enters either an uppercase or lowercase letter 'Q'. **See step #18.**

13) Define the function getComputerChoice that generates a random number representing the computer's choice and returns this value. Call this function from the main function **inside the "while loop"** and run your program to see if it works without errors. **See step #18.**

14) Define the function determineWinner that accepts the user's choice and the computer's choice (the random number) and returns a string variable representing the winner (ie. Player, Computer, Tie). Call this function from the main function **inside the "while loop"** and run your program to see if it works without errors. **See step #18.**
IMPORTANT NOTE: Remember that the user's choice is a string and the computer's choice is an integer.

15) Create a function to keep track of how many times the user wins. Create a second function to keep track of how many times the computer wins. Create a third function to keep track of how many times there is a tie. Call these functions from the main function after the determineWinner function is called, and return these "counters" to the main function inside the while loop. Run your program to see if it works without errors. **See step #18.**

16) Create two functions: one to convert from the player's choice and the other to convert the computer's numeric choice to strings that represents whether each one chose a rock, paper,

scissors, lizard, or Spock. Call these functions from the main function **inside the "while loop"** and run your program to see if it works without errors. **See step #18.**

17) Create a function to display who chose what, who won (or if there was a tie), how many times the "player" won, how many times the "computer" won, and the number of ties. Call this function from the main function **inside the "while loop"** and run your program to see if it works without errors. **See step #18.** The output should look something like the following.

```
Player chose rock
Computer chose lizard
Player wins!


Number of Computer Wins: 6
Number of Player Wins: 1
Number of Ties: 1
```

18) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

19) Type two comment lines of equal signs to indicate that this is the end of the program.
#==========
#==========

20) Save your program as LastNameFirstNameP6A.py.

21) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6A folder.

PROGRAM 6B Arithmetic Series (20 points)

Use recursion only to generate an Arithmetic Series of the numbers from the user's choice of a starting number (the 1st term) to the user's choice of an ending number (the last **term number**) with running sums of the numbers. The user should also be able to choose what should be added to each previous term to get the next term. We called this the Common Difference in your math class. Use a recursive function named **"sumUp"** to do the work. **Do not use the code from the textbook on page 212. DO NOT use iterative loops of any kind (you cannot use: "for" or "while" loops anywhere in this program)!** You will need an "if statement" to ensure the recursion has a well-defined stopping state. Print out the results in a nice table format as illustrated below using format strings and operators (Page 85-89).

Term Number	Term	Sum
1	1	1
2	3	4
3	5	9
4	7	16
5	9	25

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) Type a one line comment of all equal signs.
   #==========

4) Define the **main** function. The main function should declare and initialize all of the variables (except any variables that are better left as local variables within other functions) and call other functions. The last line of code in the main function should be:
   input("Press enter to quit the program")

5) Initialize all of the variables that are to be used in this program **in the main function** and call all of the other functions from the main function. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

6) After the main function, have a one line comment of subtraction signs. **Include this comment after each function as well.**
   #----------

7) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function and run your program to see if it works without errors. **See step #13.**

8) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does, a "type contract", and some sample function calls with sample expected outputs.**

9) Define a function that asks the user for the **first term** in the series (starting value) and returns this value. Pass the variable you declared in the main function to this function when calling it from the main function and return this variable. Call this function from the main function and run your program to see if it works without errors. **See step #13.**

10) Define a function that asks the user for the **last term number** in the series (how many terms will be displayed) and returns this value. Pass the variable you declared in the main function to this function when calling it from the main function and return this variable. Call this function from the main function and run your program to see if it works without errors. **See step #13.**

11) Define a function that asks the user for the **common difference** for the series (what will be added to each previous term to get the successive term) and returns this value. Pass the variable

you declared in the main function to this function when calling it from the main function and return this variable. Call this function from the main function and run your program to see if it works without errors. **See step #13.**

Define a "displayHeader" function to display the titles "Term Number", "Term", and "Sum" using format strings and operators (See page 85-89).
Call this function from the main function and run your program to see if it works without errors. **See step #13.**

12) Define the function "sumUp" which uses recursion (**no for or while loops allowed**) increments the term number, calculates the term and the running totals (sum) and prints the information formatted nicely using format strings and operators. Call this function from the main function and run your program to see if it works without errors. **See step #13.**

13) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

14) Type two comment lines of equal signs to indicate that this is the end of the program.
#==========
#==========

15) Save your program as LastNameFirstNameP6B.py.

16) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6B folder.

PROGRAM 6C What's The Password? (40 points)

Write a program that asks the user for the length of the passwords they want to generate and how many passwords that they wish to generate. Next, ask and then ask the user if they want to use
1) UPPER and lower case letters only,
2) UPPER and lower case letters and numerals (numbers), or
3) UPPER and lower case letters and numerals (numbers) and symbols.
Just like we did with the lottery program, save these passwords to a text file as well as displaying them on the screen. This program needs to use functions, string concatenation and lists. Use a menu as pictured below so that the user enters a 1, 2, 3, or Q (or q). **DO NOT NAME ANY FUNCTIONS MENU_CHOICE 1, etc. Function names should describe WHAT the function does.**

```
Password Generator Main Menu
1. Upper and Lower Case Letters
2. Upper and Lower Case Letters with numbers
3. Upper and Lower Case Letters with numbers and symbols
Q. Quit
```

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program**

description.

2) Leave a blank line after the DocString.

3) import random.

4) Leave a blank line after the import statement.

5) Type a one line comment of all equal signs.
#==========

6) Define the **main** function. The main function should declare and initialize all of the variables (except any variables that are better left as local variables within other functions) and call other functions.
Open a file named "passwords.txt" that will allow you to write data to this text file.
Call the function displayMyInfo.
Call the function getPasswordLength. (See #11)
Call the function getNumberOfPasswords (See #12)
Create a while loop that will continue until the user enters a lowercase "q" or an uppercase "Q" to quit running the program. Inside the while loop, call a function named "menu" (in an assignment statement) that displays all of the menu choices for this program and gets the user's choice (See #13). Close the file in the main function after the user presses Q to quit so we can have more than one type of password saved in our text file.
The last line of code in the main function should be: input("Press enter to quit the program")

7) Initialize all of the variables that are to be used in this program **in the main function** (except any variables that are better left as local variables within other functions). Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). **Initialize lists to [ ] and make sure that the word "list" is part of the identifier name for the list (ie. wordList).** Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

8) After the main function, have a one line comment of subtraction signs. **Include this comment after each function as well.**
#----------

9) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function before the while loop and run your program to see if it works without errors. **See step #17.**

10) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does. (This program does not lend itself well to type contracts and sample function calls.) DO NOT NAME ANY FUNCTIONS WITH NAMES LIKE: menuChoice1, etc. THE NAME OF THE FUNCTION SHOULD DESCRIBE WHAT THE FUNCTION DOES,**

NOT HOW IT RELATES TO THE MENU.

11) Define the function getPasswordLength that asks the user for how many characters they want in each password (int) and returns this value. Pass the appropriate variable to this function when calling it from the main function. Call this function from the main function after you open the text file and before the while loop. Run your program to see if it works without errors. **See step #17.**

12) Define the function getNumberOfPasswords that asks the user for how many passwords they want to generate (int) and return this value. Pass the appropriate variable to this function when calling it from the main function. Call this function from the main function after you open the text file, after the function getPasswordLength and before the while loop. Run your program to see if it works without errors. **See step #17.**

13) Define the function "menu" that displays the menu (as pictured below) and asks the user to enter their choice and returns this choice to the main function. Upper and lower case values should work for the menu choice to quit. Pass the variable representing the user's choice, the variable for the number of characters in each password and the variable for the number of passwords to generate, and anything else you think you may need to this function when calling it from the main function.
Next, inside the menu function type all of the "if" "elif" choices for the program. For now, just place a print statement in each choice saying what choice it represents. **As you define each function, call the function from the menu function.** Run your program to see if this works without errors. **See step #17.**

```
Password Generator Main Menu
1. Upper and Lower Case Letters
2. Upper and Lower Case Letters with numbers
3. Upper and Lower Case Letters with numbers and symbols
Q. Quit
```

14) Define the function to generate the number of passwords and the length of passwords using **Upper and lower case letters**. This function should display the passwords on the screen and save them to a text file in a similar way we did with the Lottery program. Call this function from the menu function using the appropriate if, elif choice. Run your program to see if it works without errors. **See step #17. DO NOT NAME ANY FUNCTIONS MENU_CHOICE 1, etc.**

15) Define the function to generate the number of passwords and the length of passwords using **Upper and lower case letters with numbers**. This function should display the passwords on the screen and save them to a text file in a similar way we did with the Lottery program. Call this function from the menu function using the appropriate if, elif choice. Run your program to see if it works without errors. **See step #17. DO NOT NAME ANY FUNCTIONS MENU_CHOICE 1, etc.**

16) Define the function to generate the number of passwords and the length of passwords using **Upper and lower case letters with numbers and symbols**. This function should display the passwords on the screen and save them to a text file in a similar way we did with the Lottery program. Call this function from the menu function using the appropriate if, elif choice. Run

your program to see if it works without errors. **See step #17. DO NOT NAME ANY FUNCTIONS MENU_CHOICE 1, etc.**

17) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

18) Type two comment lines of equal signs to indicate that this is the end of the program.
```
#=======================================
#=======================================
```

19) Save your program as LastNameFirstNameP6C.py.

20) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6C folder.

PROGRAM 6D Functionally I'm Sort of Searching for Monty Python (35 points)

Write a program **using functions** to practice the sorts and searches that we need to cover. This program will practice the Python list methods as well as the standard sorting and searching algorithms. **DO NOT USE ANY BREAK STATEMENTS IN YOUR PROGRAMS.**

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) import random

4) Leave a blank line after the "import" statement.

5) Type a one line comment of all equal signs.
```
#============================================================
```

6) Define the **main** function. The main function should declare and initialize all of the variables including the two lists (except any variables that are better left as local variables within other functions) and call other functions.
Create a constant LOW and initialize it to 1 (one).
Call the function displayMyInfo.
Create a while loop that will continue until the user enters a lowercase "q" or an uppercase "Q" to quit running the program. Inside the while loop, call a function named "menu" (in an assignment statement) that displays all of the menu choices for this program and gets the user's choice. Next, call a function named controlMenuExecution passing the parameters menuChoice, originalList and listCopy. Please use these names for your lists as the word "list" is a reserved word in Python. **Use originalList only for menu choice 1. Use both originalList and listCopy**

for menu choice 2 to copy the list. Use listCopy for all of the other functions and menu choices.
The last line of code in the main function should be:
input("Press enter to quit the program")

7) Initialize all of the variables that are to be used in this program **in the main function** and call the functions listed in step #6 from the main function. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). **Initialize lists to [ ] and make sure that the word "list" is part of the identifier name for the list (ie. originalList).** Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code".

8) After the main function, have a one line comment of subtraction signs. **Include this comment after each function as well.**
#----------

9) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function before the while loop and run your program to see if it works without errors. **See step #21.**

10) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does. (This program does not lend itself well to type contracts and sample function calls.) DO NOT NAME ANY FUNCTIONS WITH NAMES LIKE: menuChoice1, etc. THE NAME OF THE FUNCTION SHOULD DESCRIBE WHAT THE FUNCTION DOES, NOT HOW IT RELATES TO THE MENU.**

11) Define and implement the function named **menu** which prints all of the menu choices, inputs and **returns** the user's choice. **(Don't forget to call it in an assignment statement from the main function.)** Include a comment line of subtraction signs after the function. Call this function from the main function before the while loop and run your program to see if it works without errors. **See step #21.** Display the menu (as pictured below) and ask the user to enter their choice.

```
 I'm Sort Of Searching For Monty Python Part 2

1.  Generate Random Numbers For The Original List
2.  Copy The Original List
3.  Sequential (Linear) Search
4.  Binary Search Iteration (The List Must be Sorted)
5.  Binary Search Recursion (The List Must be Sorted)
6.  Selection Sort
7.  Bubble Sort
8.  Insertion Sort
9.  Quick Sort
P.  Python's Sort Method
S.  Python's Search and Count Method
I.  Insert an element in the List at any position
D.  Delete an element from the List at any position
0.  Display the numbers using a 'loop' (press zero)
Q.  Quit
```

12) Create a function ControlMenuExecution which accepts menuChoice, originalList, and listCopy as parameters and contains the "if, elif, else statements" corresponding to the menu choices which calls the other functions. **Declare logicalSize as a local variable in this function.** Call this function from the main function inside the while loop, after the menu function is called and run your program to see if it works without errors. **See step #21. (**Until you get all of the functions implemented you can use print statements saying that this choice has not been implemented yet to test the if, elif statements.)

13) Create a function getLogicalSize which accepts logicalSize as a parameter and returns logicalSize to ControlMenuExecution. Pass the variable logicalSize to this function when calling it from the ControlMenuExecution function. In this function for menu choice #1, ask the user how many elements they want in the list. Call this function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

14) Create a function generateRandomNumbers which accepts originalList and logicalSize as parameters and fills the list with random numbers from 1 to logicalSize (use the constant LOW for 1). Call this function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

15) Create a function which accepts originalList and listCopy as parameters and copies the originalList to listCopy. **The first line of code in this function should apply the clear() method to listCopy before copying the originalList by appending the elements.** If you don't do this, the list will double in size every time we call menu choice 2. **Use a "for loop" to append the contents of originalList to listCopy to copy the list. Do NOT copy the list using "slicing" ie. listCopy = originalList[:] , when using functions.** Slicing will create a local copy of listCopy in this function which makes **listCopy** an empty list for the rest of the program. Call this function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

16) Create a function which accepts listCopy as a parameter and displays the contents of listCopy using a "for loop" to display 10 numbers across each row. Call this function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21. Run your program with menu choices 1, 2, and 0 (zero) in that order to make sure that everything works so far.**

17) For this program we will create a separate function to implement each menu choice. There are PowerPoint lessons for all of the sorts and searches on my website. Upper and lower case values should work for the menu choices that use letters. Write each function one at a time and call it from ControlMenuExecution to see if it works. **See step #21.**
DO NOT NAME ANY FUNCTIONS WITH NAMES LIKE: menuChoice1, etc. THE NAME OF THE FUNCTION SHOULD DESCRIBE WHAT THE FUNCTION DOES, NOT HOW IT RELATES TO THE MENU.

18) For both of the Binary Searches, we are determining presence or absence of the "target" number. For the Linear Search, be sure to display the index numbers of every occurrence of the "target" number.
The Python search, menu "S", should display the index numbers of every occurrence of the "target" number as well as how many occurrences there are of the target. Use the Python count method for this.
Ask the user what number they are looking for before calling each function. Don't forget to account for the fact that the number that I search for when I test your program may not be in the list. If this is the case, please add code that says that the number was not in the list for each of the searches listed above. As you write each function, call the function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

19) In the function to insert an element into the list, ask the user where (what index number) they would like to insert the number and then what number they wish to insert (don't error trap the number the user wishes to insert, any number is fine).
For the function that deletes an element from the list ask for the index number of the element that the user wishes to delete. As you write each function, call the function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

20) Create a separate function for each of the sort methods (including the Python sort method). As you write each function, call the function from the ControlMenuExecution function and run your program to see if it works without errors. **See step #21.**

21) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

22) Type two comment lines of equal signs to indicate that this is the end of the program.
#==========
#==========

23) Save your program as LastNameFirstNameP6D.py.

24) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6D folder.

Here is the menu again.

```
 I'm Sort Of Searching For Monty Python Part 2

1.  Generate Random Numbers For The Original List
2.  Copy The Original List
3.  Sequential (Linear) Search
4.  Binary Search Iteration (The List Must be Sorted)
5.  Binary Search Recursion (The List Must be Sorted)
6.  Selection Sort
7.  Bubble Sort
8.  Insertion Sort
9.  Quick Sort
P.  Python's Sort Method
S.  Python's Search and Count Method
I.  Insert an element in the List at any position
D.  Delete an element from the List at any position
0.  Display the numbers using a 'loop' (press zero)
Q.  Quit
```