Computer Science 220L
Laboratory 10 – `while` Loops & List Methods

Learning objectives:

- Develop `while` control structures.
- Use Python's built-in list methods.
- Perform linear search on data.

Do NOT use `for` loops in this lab. Use `while` loops. Demonstrate each exercise for an instructor as you complete it.

1. Using Python's built-in list functions

Let's learn a bit by doing. Lists, as you know are objects and have methods. We have seen one method already, append(). Below is the Python documentation for two other methods.

`list.remove(`*x*`)`

Remove the first item from the list whose value is equal to *x*. It raises a `ValueError` if there is no such item.

`list.pop([`*i*`])`

Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

Write a void function, `findAndRemoveFirst(list, value)` that modifies a list by finding the position of the first occurrence of `value` and then inserts your name in that position. It should then remove the first occurrence of `value.` (Try removing `value` two ways with `pop()` and `remove()`.) Add code to `main()` to test your function making sure to pass a value that is in the list and a value that is not in the list when you test. You can find the item manually.

Add code to `main()` to test `findAndRemoveFirst().`

2. Linear Search

The file `dataSorted.txt` contains series of numbers in sorted order on multiple lines of a file. **Each line may contain one to many numbers.** Write a function `readData(filename)` that accepts the name of a file, builds a list of the data, and returns the list. Write another function `foundPosition(searchVal, values)` that accepts a value and a list of comparable values, and returns an integer representing the position of the searchVal in values. It should return `-1` if searchVal is not in the list. Your code should perform an efficient, elegant, and manual linear search. Add code to `main()` to test your functions. Remember, no for loops or breaks allowed.

Add code to `main()` to test `readData()` and `foundPosition()`.

3. Forcing good input

Often a program needs an input within a certain range of values. Now that we have indefinite loops we can force the user to enter a value in the correct range. Simply ask for the input again and again until it is in the correct range. Note: It is very important to tell the user why you are asking for the input again. Otherwise, repeated "this is wrong" messages are annoying and frustrating for the user of the program.

Write a function called `goodInput()` that asks for a number to be input that is in the teens or in the fifties. If the input is outside that specification, ask for input again. The function should return the "good input" received by the user.

Add code to `main()` to test `goodInput()`.

4. Counting the digits in a number

Write a function called `numDigits()`. This function repeatedly asks the user to input a positive integer. End the function when a zero or negative number is entered.

For each number entered, the function should print the number of digits found in that number. You could easily answer this question by reading the number as a string and using `len(number)`, but that wouldn't teach you about `while` loops.

Integer divide the number repeatedly by 10 until the number reaches zero, and count the number of divisions. That number is the number of digits.

Add code to `main()` to test `numDigits()`.

5. High-Low game

Write a function called `hiLoGame()` that provides a number-guessing game.

Generate a random number between 1 and 100 to be guessed. Write a loop that allows the user seven guesses. For each guess, the game tells the user if the guess was "correct", "too high", or "too low." The game continues until the user guesses correctly or has guessed incorrectly seven times. At the end of the game, one of the following messages is displayed: "You win in # guesses!" or "Sorry, you lose. The number was ##."

To generate a random number, put `from random import randint` at the top of the program. Then, for example, a call to the function `randint(1,10)` generates a pseudo-random number in the range [1..10], i.e., including both 1 and 10.

Add code to `main()` to test `hiLoGame()`.

Upload the file (don't forget to submit).