1. **zyBooks Labs**

Please follow the link on Canvas to complete the following zyBooks labs:

* + 12.8 *LAB: Words in a range (lists)*
  + 12.9 *LAB: Word frequencies (lists)*
  + 12.10 *LAB: Sorting TV Shows (dictionaries and lists)*
  + 16.10 *LAB: Descending selection sort with output during execution*

This portion of the lab will be worth 50% of your Lab 11 grade.

1. **Linear vs. Binary Search**

In this component of the lab you will write a small, but complete Python 3 program called **Lab11A.py** that measures the time it takes to search a very large list using a linear and then binary search algorithm.

* 1. Create a user-defined function called linear\_search(numbers, key) that accepts two string arguments, a list of integers and a integer key to search for in the list. You may use the linear\_search() function defined in the lecture notes.
  2. Create a user-defined function called binary\_search(numbers, key) that accepts two string arguments, a list of integers and a integer key to search for in the list. You may use the binary\_search() function defined in the lecture notes.
  3. In the main part of the program, create an empty list.
  4. Then, in a loop of your choice, generate 10,000,000 random integers from 1 to 100,000,000, inclusively, and append to your list. Note that we will not be printing out the list as the size will be too large, but if you want to verify that it is generating the integers correctly, do so for 100 integers in your list as an initial test.
  5. Add the following line to the top of your program to use measure time in a human-readable format:

from datetime import datetime

* 1. Call the datetime.now() function to get the current time, assigning the result to a variable called start\_time.
  2. Since we want to measure the performance of the linear search versus a binary search, we need to ensure that it searches the entire list. Since our list contains integers from 1 to 100,000,000, we can simply search for any negative integer, so call your linear\_search() function, passing your list and some negative integer.
  3. Call the datetime.now() function to get the current time, assigning the result to a variable called end\_time.
  4. Print out the search time as a difference of end\_time and start\_time and observe the amount of time it took to do a linear search.
  5. Before we can use our binary\_search() function, our list has to be sorted, so use the sorted() function to sort your list. The sorting operation should actually take most of the time it takes for your program to run.
  6. Call the datetime.now() function to get the current time, assigning the result to a variable called start\_time.
  7. Call your binary\_search() function, passing your list and some negative integer so that it searches through the entire list.
  8. Call the datetime.now() function to get the current time, assigning the result to a variable called end\_time.
  9. Print out the search time as a difference of end\_time and start\_time and observe the amount of time it took to do a binary search. You should notice that the binary search took significantly less time than the linear search to go through the entire list.

For example, the output might look like this (input shown in **bold**):

$ **python3 Lab11A.py**

Linear Search Results: -1

Linear Search Duration: 0:00:00.529240

Binary Search Results: -1

Binary Search Duration: 0:00:00.000084

*Note that you will submit this file to Canvas.*

Now that you have completed this lab, it’s time to turn in your results. Once you've moved the files to your windows machine (using **WinSCP**), you may use the browser to submit them to Canvas for the **Lab 11** **BONUS** dropbox.

You should submit the following files:

* **Lab11A.py**
* **(Note that the zyBooks labs are submitted separately through Canvas.)**

Ask your TA to check your results before submission.

Now that you've finished the lab, use any additional time to practice writing simple programs out of the textbook, lectures, or even ones you come up with on your own to gain some more experience.