COMSC-205PY FALL 2020 ASSIGNMENT 5

Due November 30th at 9:00am Eastern Time

(Recommended submission date: November 25)

Submission checklist

Your final submission should include the following

- **DataSearch.py**

At the top of each file, clearly state:

- Your name and the name of the assignment
- Attribution for any sources used, including people (other than instructors) who you asked for help
- Description of the file

Read all the way through this assignment twice before you start working on it. The second time through, take notes, plan out how you will work on this over the next 6 days, and work on a design for the program.

Background

In this assignment, your code will mimic a database search. The program will use two sets of integer values. **seek** refers to the searched-for values (values we want to look up), and **data** refers to the database values (values we want to search in). At run time, the seek and data values will be in two different files. The names of both files **must** be provided on the command line when the program is executed. You can look back at Assignment 4 to remind yourself of how to pass in command line arguments. The first argument is in **sys.argv[1]**, the second argument is in **sys.argv[2]**, and the third (relevant below) is in **sys.argv[3]**.

After the information is read in, a binary search will be used to determine if any given seek value exists in data. In order for binary search to work, data must be sorted, least to greatest. The program will then write what values were found (and what were not) to an output file, the name of which is also given on the command line.

Writing output to a file

To write to a file in Python:

- Open the file in write mode (for input you opened a file in read mode). For example, if the output file is `results.txt` and you want to refer to the file stream as `fo`, then use the command

```
fo = open(sys.argv[3], "w")
```

- Within the program, refer to the output stream using `fo`, the **file handle**. Rather than using print( ) statements which output to the screen, use the write method which will put its argument into the output file.

```
fo.write("Here is the output from my program")
fo.write("x + y give us ", x + y)
```

- As with an input file, before the program ends, close the output file

```
fo.close()
```

The Assignment

Your program should do the following:

1. Using the input files specified on the command line:
    a. Read the seek values into a list
    b. Read the data values into a different list
2. Create a third list to keep track of *found* information
3. Sort the data list
4. Look up each seek value in the data list, using a ***recursive binary search***
5. Record in *found* an indication of whether or not the seek value was found:
    a. if seek[i] is found, then found[i] should be set to 1
    b. otherwise set found[i] to 0
6. In a "human readable" form, write to the designated output file (also specified on the command line) each value of seek and the corresponding value from found.

Input Management

The program should take in three command line arguments: a seek file name, a data file name, and an output file name, in that order. First, the seek values should be copied into a list. Then, the data file should be read in the same way. The data list should be sorted, and a third list should be created to keep track of whether the searched-for numbers have been found.

Binary Search

Write a recursive binary search function. For each value in seek, search through the data to determine if it's present or not. Use the found list to keep track of whether each number is found. You can find information on binary search in section 6.4 of the textbook.

Output

Once the search is complete, the searched-for values should be written to a file (see above). This should be in a human-readable format: each number should be printed to the file, followed by "Yes" (if it was found) or "No" (if it wasn't). Both the seek and found lists will be used for this.

Comments and Coding Style

Make sure every function you write is documented and that your code is well-organized.

How to Approach the Assignment

1. Binary search is the most valuable part of this assignment, so it's useful to work on first. While binary search can be implemented both recursively and non-recursively, for this

assignment, your solution **must be recursive**. Similarly, there are multiple ways to keep track of the portion of the list that your code is currently searching within. Make sure to name all parameters and variables clearly so you can keep them straight!

2. Start with some hard-coded ***sorted*** lists so that testing is easier, then hard-coded ***unsorted*** lists, then add file reading later and comment out or delete the testing sections before submission!

3. The example files are good to test with, but we will also be using other files when running your code. What happens with your code when there are negative numbers in a file? A zero? Does it still work as expected? All files tested will separate numbers by spaces, **not by new lines**, so make sure the files you test with follow this format.

4. The focus of this assignment is on binary search and reading and writing from files. You do not need to define classes for this assignment, though it is okay if you choose to do so.

Sample run

When running, the program should take in filenames in this order:

```
python3 DataSearch.py seek.txt data.txt output.txt
```

If you test with the `seek.txt` and `data.txt` available on Moodle, the `output.txt` should look like this:

```
-1: No
1: Yes
4: No
15: Yes
16: No
44: No
49: Yes
69: No
```

Rubric

Each section will be given a letter grade. The average of those letters will generate your final score. Note: Moodle does not allow letters, so they will be translated into a percentage there.

	F	D	C	B	A
Input	Files are not read in; lists are hard-coded	One file is read in	Both files are read in, but some data is missing, or the order is swapped	All information from both files are read in, but not all of it is run through binary search, or data isn't sorted	All information from both files are read in, sorted, and run through binary search
Binary Search	Binary search does not exist	Binary search exists, but never returns the correct answer, or is never called	Binary search exists, but is not recursive, or is missing a base case, or loops infinitely, or has a bug that frequently gives the wrong answer	Binary search exists, but has a bug that causes some wrong answers	Binary search exists, is recursive, and returns the correct answers
Output	Results are not saved to a file	Output file is created, but nothing is written to it	Some results are not saved	All results are saved, but not in a human-readable format	All results are saved in a human-readable format
Comments, naming, style	No comments present, no name in file, method names incorrect	Name in file, some methods named correctly, no comments	Name in file, description of file in comment, methods named correctly (typos ok), more than one error that prevents running	Name in file, description of file and most methods, methods named correctly, some variables named meaningfully, significant over- or under-use of comments or one error that prevents running	Name in file, description of file and all methods, methods named correctly, all variables named meaningfully, other comments as needed, no errors that prevent running

Have you read all the way through this assignment twice? If not, read it again!