CS 51 Laboratory #11
Searching in Python

Due: Friday, April 27, at 4 p.m.,

For this lab, you will be writing two functions in Python, a program to do a linear search of a list and another to do a binary search. You will then test these and provide timing information for the run-time of these searches.

In this lab we'll work within the PyCharm integrated development environment for Python.

1. You can find PyCharm in the applications folder on your computer. Open PyCharm by clicking on the icon labelled PC. You can accept the defaults and, eventually, should get to a screen that invites you to select Create New Project. Go ahead and create the project, making sure that the location of the project is: `/home/***/Desktop/CS51GWorkspace/Lab11`, where *** is replaced by your CS username.

2. You'll now be looking at the main screen for the PyCharm IDE. For now we'll just worry about running the interpreter. If there isn't a Python Console window at the bottom of your screen, select from the View menu, the Tool Windows submenu, and select Python Console. This will open a frame at the bottom of the window which is running the Python interpreter. The interpreter gives you the prompt `>>>`. To see that Python is working properly, first type 6/4 and hit the return key. The system should respond with 1.5. Then type 6//4, return. Now you should get 1 as the // operation is integer divide.

Of course, we don't just want a calculator, we want to write a real program. We'll need to create a program file for this. Look at the navigation pane on the left of the screen and double click on Lab11 so that it shows you the contents of that project. Now right-click the first item (it should start with the an icon of a folder named venv) and create a new Python file named `lab11.py`.

1. In that new file enter the code: `print(''Hello world!'')`

2. Save the file either by selecting File Save All or by pressing fan-S .

3. Run the file by using the Run menu to select Run.... (Note that by default PyCharm tries to run the last program that you ran, so this is the procedure you'll follow the first time you run any new program. After that, you may find it easier to use the keyboard shortcut.)

4. Add a few more lines to your program so that it prints something more interesting.

If all this works you are ready to write your program.

Searching in Python The code for doing linear and binary searches can be found on-line at
`http://www.cs.pomona.edu/classes/cs051G/demos/SearchSort/search.grace`.
You can choose whether you want to do iterative or recursive versions of the searches: you need not do both.

The iterative versions are single methods: `search` and `iterativeBinarySearch`, while the recursive versions are `rsearch` and `rBinarySearch` (with associated helper functions). Transcribe these as methods in Python. Keep in mind that methods in Python are written with keyword `def`, and that the headers of functions and control blocks all end with ":". I highly recommend that you keep the Beginner's Python cheat sheet at:
`http://www.cs.pomona.edu/classes/cs051G/handouts/beginners_python_cheat_sheet_pcc_all.pdf`
open on your desk top as you work.

Here is a reminder of a very simple Python function we talked about in class:

```
def filterEvensNSquare(aList):
    answer = []
    for val in aList:
        if val % 2 == 0:
            answer.append(val*val) return (answer)
```

Test your code on input to make sure it works correctly. Remember that binary search only works if the list is sorted. If `lst` is a list, then `lst.sort` updates `lst` so that all of the elements are in order. That is, it updates the existing list rather than creating a new one.

Timing the searches In order to see the difference in efficiency between the kinds of search, I'd like you to measure the time it takes to do each of these searches. In the cs051G folder on your desktop, you'll find folder PythonSearch and within that the file timingCode.py. This file contains code to time your functions. It should be copied from that file to the end of your program.

For convenience, I have also copied the code here, though I suspect copying and pasting this code from the pdf may result in indenting issues for PyCharm that you'll have to fix.

```
import random
from timeit import default_timer as timer

def timing():
    """
    Function to perform timing on linear and binary search routines.
    We assume the linear search method has name linearSearch, while the binary search
    has name iterativeBinarySearch.  If your functions have different names, change them
    in the code.
    """

    count = 5000
    myList = []

    while count <= 1000000:
        count = count * 2
        for i in range(count):
            myList.append(2*random.randint(1,count + 1))

        myList.sort()
        elapsedBinary = 0
        elapsedLinear = 0

        print ("\nTiming search times for "+str(count)+" elements")
        for i in range(5):
            toFind = 2 * random.randint(1, count + 1) + 1
            start = timer()
            index = iterativeBinSearch(myList,toFind)
            elapsedBinary = elapsedBinary + timer() - start

            start = timer()
            index = linearSearch(myList,toFind)
```

```
            elapsedLinear = elapsedLinear + timer() - start

        print ("Average binary search time for "+str(count)+" is "+
                                str(int(1000000 * elapsedBinary/5))+" microseconds")
        print ("Linear search time for "+str(count)+" is "+
                                str(int(1000000 * elapsedLinear/5))+" microseconds")

timing()
```

The method timing() will execute linear searches and binary searches on files of size: 10000, 20000, 40000, up to 1280000 where successive sizes are double their predecessor. For each of these sizes a list of random numbers is generated and then search times for five randomly chosen elements (named `toFind`) are computed and the average printed for both binary search and linear searches. The randomly selected items are chosen to be numbers NOT in the list. WHY?

In a block comment at the end of your program, please copy the output of this method and forecast what would happen to the times for each of binary and linear searches if we continued doubing the size of the list. Note that any background computation on the computer will affect the timing results. Hence you should not expect perfectly predictable timing results. In particular, the binary search times are so small that you will see a lot of random fluctuations

Turning it in When your work is complete you should deposit in the dropoff folder. Before turning it in, please be sure that your folder contains all of the .py files, and make sure it will run with no modifications necessary on our part. If we have to make modifications to get it to run, it will cost you points, even if it is just changing the name of files.

Before turning it in, make sure the folder name has the form `lab11_lastnamefirstname`. Also make sure to double check your work for correctness, organization and style. In particular, the comment at the top of each file MUST include your name.

Table 1: Grading Guidelines

Value	Feature
3 pts.	linear search
3 pts.	binary search
2 pts.	Analysis of search times
2 pts.	Miscellaneous code quality