**Computer Science 220L**
**Laboratory 11 –List Methods and Searching/Sorting**

**Learning objectives:**

- Perform binary search on data and compare search/sort algorithms
- Make good use of functions
- Solve a real-world problem with all of the knowledge and skills you've gained!

**0. Linear Search**

During lab 10, you wrote a `readData()` function. A working solution to that is in the `algorithms.py` folder. Write another function, similar to `foundPosition()` from last week. Implement, `isinLinear(searchValue, values)` → `Boolean`. This function accepts a `searchValue` and a list, it returns `True` if the `searchValue` is in `values` and `False` if it is not. Your search should be manual and efficient. You can**not** assume that the data found in values is sorted. Add this function to the `algorithms.py` file.

**1. Binary Search**

Now add a function `isInBinary(searchVal, values)` to `algorithms.py` that works the same way as `isInLinear()` but instead performs a binary search of the data. Add code to Lab11.py's `main()` to test your function. You will need to import your `algorithms` file; for this to work, algorithm.py and Lab11.py should be in the same director. `from algorithms import *`

**2. Sorting**

Add a function `selectionSort(values)` to `algorithms.py`, that accepts a list of values and sorts them in place using the selection sort algorithm discussed in class.

The selection sort is described fully in your text. It works by repeatedly: 1. Finding the position of the smallest value in the unsorted list and 2. Swapping this value with the front of the unsorted portion of the list.

**3. Comparative analysis of sorts**

Download and run the program `sortTest.py`. This code sorts a small list and a large list using your sort and Python's built-in list method `sort()`. Make observations about the output as part of your header comment for `algorithms.py`.

**4. Sorting rectangles**

Add a function `rectSort(rectangles)` to `algorithms.py`, that accepts a list of Rectangle objects and sorts these objects based on the area of the Rectangles using the selection sort algorithm you wrote. You should write a `calcArea(rect)` function to help. Add code to `main()` to test your function.

*Pick one of the two "capstone" problems below:*

**5. Capstone Option 1**

*Poof* You're now a professional investor! What you're interested in is identifying bursts of activity in the stock market very quickly so that you can jump on opportunities (or run away from danger). The file trades.txt contains a whole bunch of numbers. All the numbers together represent the trading volume of a particular stock on a particular hour. Each number

represents how many times this stock was traded in a one-second interval, so there is one number for each second in the hour.

Your goal is to write a program `tradeAlert(fileName)` that alerts us with a warning if the trading volume exceeds 830 and alerts us with a message to pay attention if the volume equals 500 at any second. Be sure to include timestamps that tell us when exactly during the day (in seconds) these messages are triggered. Take advantage of the functions you already have available to you!

**6. Capstone Option 2**

*Poof* You're now an astrophysicist looking for neutron stars! Neutron stars are interesting for all kinds of reasons I don't understand, but we want to find one. The key to finding one is that some of them occasionally emit strong bursts of energy in our direction, aptly called the lighthouse effect. So if we point our research satellite in a particular point and wait for a little while, we would hope to find a recurring strong signal in and among all the other noise we're getting from all the other celestial objects.

Write a program `starFind(fileName)` that alerts us if we've found the neutron star. The file signals.txt contains a whole lot of numbers. These numbers represent the magnitude of signals received by our satellite over a period of time. What you're looking for is a recurring strong signal. If you find at least five signals that fall in the range 4000-5000, then you've found a new neutron star! Be sure to save and report ow many pulses you found, the strength of each signal, and how many signals you had to search before you found your fifth signal.