# CSC 110 – Lab 7
# Algorithm Analysis – Sorting and Searching

Names: ________________________________________

## *Introduction*
The purpose of this lab is to practice analysis of sorting and searching algorithms.

## *Searching*

In class we discussed the Binary Search algorithm as a better alternative to using a Sequential (Linear) Search.

Under what circumstances will the Binary Search algorithm work?

When these circumstances apply, why Binary Search better than Linear Search.

Binary Search Applet – How it works

Now we will use a simple applet to visualize how Binary Search works. Open this Binary Search applet.

http://euler.slu.edu/~goldwasser/demos/BinarySearch/

- Start with a list of 8 items (enter 8 into the box labeled N).
- Enter 1200 for the width of the window and 800 for the height of the window.
- Enter 0 for the Seed value.
- Enter 18 in the key box. This is the number that you are searching for.

Click on the "run" button to execute the search. A new window will appear. How many comparisons did it take to find the number in the list?

Close the pop-up window by clicking on the File menu and choosing "Close".

Now search the list for the number 1 (put 1 in the key box). How many comparisons does the algorithm have to do to compute the result?

Instructor Initials: ______________________

Now create a list with N=32 items using a Seed value of 1. How many comparisons would you expect the algorithm to do to search for a number that is not in the list? Explain.

Test out your theory by searching this list for the number 1. Were you correct?

How many comparisons would the Linear Search have to do to perform this same search?

What if your list had 16384 items - how many comparisons would Binary Search have to do to search for a number that is not in the list?

What about Linear Search?

Instructor Initials: ______________________

Complete the algorithm

The following file contains an almost complete Python function to do the binary search that we discussed in class – searching a phone book for a name and returning the phone number.

http://www.cs.uri.edu/~cingiser/csc110/labs/python/binary_search_skeleton.py

Most of the code is written for you already. Fill in the rest of the code and test it out with the data in this file:

http://www.cs.uri.edu/~cingiser/csc110/labs/python/phones.txt

Make sure the function works for names in the list as well as names that are not in the list.

Instructor Initials: ______________________

***Sorting***

In class we spent some time discussing the Selection Sort algorithm. Here we will consider several other sorting algorithms and compare them based on how much work they do.

Open the following web page:

http://math.hws.edu/TMCM/java/labs/xSortLabLab.html.

The applet on this page will allow us to visualize how several different sorting algorithms work. Click on the button labeled "Launch xSortLab" to run the applet.

Describe the algorithms:

For each of the following sorting algorithms, run the applet and describe how the algorithm works to sort a list of numbers:

*Bubble Sort:*

*Insertion Sort:*

Instructor Initials: ____________________________

<u>Consider the Python code:</u>

You can find Python code for each of the algorithms here:

`http://www.cs.uri.edu/~cingiser/csc110/labs/python/sort.py`

Open this file in IDLE and run the code. Test out the sorting algorithms by typing:

```
insertionSort([2,5,3,4,1])


bubbleSort([2,5,3,4,1])
```

Instructor Initials: ____________________________

<u>Analysis of the algorithms:</u>

In order to compare the performance of these algorithms, we need to count how much work is done by each. For sorting algorithms, the important units of work are comparisons of numbers and swapping of values in the list.

For each of the algorithms, add a few lines of code to the Python functions to count the number of comparisons and the number of swaps done by the algorithm.

Now we will examine how much work is done by each algorithm in the best case and the worst case.

In general, what is the best case input for a sorting algorithm?

What is the worst case input?

In order to get a good idea of the performance of an algorithm, it should be examined on larger data sets. Download the following data files to be used in this analysis:

*List of numbers sorted:*
http://www.cs.uri.edu/~cingiser/csc110/labs/python/data_sorted_100.txt

*List of numbers in reverse (descending) order:*
http://www.cs.uri.edu/~cingiser/csc110/labs/python/data_reverse_100.txt

You will notice in the sort.py file that you downloaded, there is a function to get the data from a file and store it in a list. You will use this function to read the data from the data files above, and run the sorting algorithms for analysis.

In the Python shell, use the following commands to run bubble sort on the sorted data file:

```
myList = getList('data_sorted_100.txt')

bubbleSort(myList)
```

This will display the sorted list along with the number of comparisons and swaps that you counted in the previous step.

Run each of the algorithms on the two data files provided (best case and worst case). Each time you run an algorithm, you will have to run the getList function get the unsorted list. Describe your results in terms of the number of comparisons done by each algorithm and the number of swaps done:

*Insertion sort:*

- Best case:
  - Comparisons:
  - Swaps:
- Worst case:
  - Comparisons:
  - Swaps:

*Bubble sort:*

- Best case:
  - Comparisons:
  - Swaps:
- Worst case:
  - Comparisons:
  - Swaps:

Instructor Initials: ______________________________

***Challenge Problems***

1) Merge sort is another sorting algorithm that you can find in xSortlab applet. Run the applet and describe how this sorting algorithms works to sort a list of numbers.

Instructor Initials: ______________________________

2) Write a Python function that executes the merge sort. Demonstrate that it works on a list of at least 10 numbers.

Instructor Initials: ______________________________