Exercise 1

Name Shakir Ali Date 2015/10/22

Section _____

1) Start the "Sorting" applet.
2) Make a table in which you can record the performance of the three algorithms on the four different data sets:

	Ex 1	Ex 2	Ex 3	Ex 4
selection	28	120	120	120
bubble	27	75	15	120
quicksort	26	66	150	132

3) Run the applet 12 times and record the numbers it generates in the big white text area.

recorded above

4) Which algorithm is usually the fastest?

bubble sort

5) Is any one algorithm always faster than the others?

bubble sort is always faster

6) Are any two algorithms similar?

bubble and quick sort are similar

7) Did the four datasets cover all the major "cases" you can think of, or are there others? If so, describe them.

yes there is one more data set called almost sorted which has groups of values unsorted with increasing values of groups

Exercise 2

Name _____ Date _____

Section _____

1) Let's experiment with datasets other than those in the examples. Start the "Sorting" applet.

2) In the text area where it says "Your data set goes here," remove the text and type in these numbers. Press return after every number except the last.

```
1
99
2
98
3
97
4
96
5
95
6
94
7
93
8
```

3) Are these numbers sorted? Are they arranged randomly? What will the final sorted list look like?

These numbers are not sorted, and not randomly generated. final list looks like having adjacent integers

4) Now let's see how our three sorting algorithms perform. Run each of the three algorithms and write down how many comparisons they required to finish the job. (Note: you don't have to retype the numbers in after each sort because the applet refreshes the numbered list on the left with the values you typed into the big text area but does not change the big text area.)

Selection sort 105

Bubble sort 105

Quicksort 73

5) Which is the fastest? Does that surprise you?

Quick sort is fastest. it surprises me yes

6) Rerun the quicksort algorithm and write down any patterns you see occurring in the numbered list to the left. For instance, maybe the algorithm quickly moves all of the larger numbers to the top or bottom section.

quick sort is good for already sorted alternate indices of an array

Exercise 3

Name _____ Date _____

Section _____

1) Now for yet another dataset. Start the "Sorting" applet.

2) In the text area where it says "Your data set goes here," remove the text and type in these numbers. Press return after every number except the last.

```
1
3
8
15
26
45
50
97
92
84
79
72
70
68
51
```

3) Are these numbers sorted? Are they arranged randomly? What will the final sorted list look like?

4) Now let's see how our three sorting algorithms perform. Run each of the three algorithms and write down how many comparisons they required to finish the job.

   Selection sort _____

   Bubble sort _____

   Quicksort _____

5) Which is the fastest?

6) Rerun the quicksort algorithm and describe what happens to the number 97.

7) How is quicksort's behavior on this dataset different than its behavior on the previous dataset?

Exercise 4

Name _____ Date _____

Section _____

1) Start the "Searching" applet. The dataset in the numbered list on the left is fixed—you can't change it. However you can select an algorithm and a value to search for.
2) Run all four searches that are in the examples and write down how many comparisons were made:
   _____ Example 1: Something in the list (sequential)
   _____ Example 2: Something not in the list (sequential)
   _____ Example 3: Something in the list (binary)
   _____ Example 4: Something not in the list (binary)
3) Select "binary search" as your algorithm and try to find 62. How many tries does it need to find it?

4) Try to find 62 using sequential search. How many tries does it need?

5) Which searching algorithm is clearly superior on this dataset?

6) Is the numbered list on the left sorted? Is this a prerequisite for binary search? Is it a prerequisite for sequential search? (Hint: see p. 306 of your textbook.)

Exercise 5

Name _____ Date _____

Section _____

1) Start the "Searching" applet. Pull down the algorithm menu so that "Binary search" is chosen.

2) This may be a little tedious but there's a reason, so bear with it. For each value in the numbered list to the left, try to find it. Type it into the top text area and click the Search button. Then write down how many tries it required to find it.

   5 _____
   16 _____
   19 _____
   25 _____
   37 _____
   44 _____
   56 _____
   62 _____
   79 _____
   81 _____
   99 _____
   100 _____
   105 _____
   117 _____
   200 _____
   300 _____

3) What is the greatest number of tries needed? _____

   What is the least number needed? _____

   What number of tries appears most frequently in your result list? _____

4) There are 16 values in the list that is searched. Suppose you had 32 values. What do you guess would be the most frequent number of searches? _____

   What if your list had 64 values? _____