*INFO-3135, F2025*

Week 4 Practice Exercise

**From the Sort Algorithms Lecture and Demonstration**

Use the slides posted in Week 4 to answer the following questions:

1. The insertion sort algorithm uses what technique?
2. The bubble sort algorithm uses what technique?
3. The selection sort algorithm uses what technique?
4. Given the following list:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 90 | 8 | 653 | 56 | 123 | 1 | 235 | 7 |

Show a trace of execution for:

1. Selection sort
2. Insertion sort
3. Bubble sort
4. Give examples of these cases, using a data set where size = 6:
5. Worst case
6. Best case
7. Average case

**Coding Practice**

1. Return to the sort demo in Week 4 called SortUsingIteration. Write a second version of each of the iterative sorts using while loops rather than for loops.

* Bubble Sort, Insert Sort, Selection Sort

1. Return to your Week 2 Lab Exercise (*handout is posted in Week 2*). Add the following sort functions to your array-implemented Vector class:

* ascendingSelectionSort()
* descendingInsertionSort()
* ascendingBubbleSort()

Return to the tester file, and add the following code to add 100 random values to the vector:

|  |
| --- |
| #include <cstdlib>  #include <ctime>  //add to main  //seed random  //populate the vector with 100 random values between 1-100  srand(static\_cast<unsigned>(time(nullptr)));  for (int i = 0; i < 100; ++i)  v.push\_back(rand() % 100 + 1); |

Test each of your sort functions; call each sort() and print().

1. Return to your LinkedList implementation (from Week 3). Add the following sort functions to your LinkedList class. You will need to use while loops and a temporary Node\* to traverse the list:

* ascendingSelectionSort()
* descendingInsertionSort()
* ascendingBubbleSort()

Return to the tester file and add the code from above to add 100 random values to the LinkedList. Test each of your sort functions; call sort() and print().

1. Return to your DoublyLinkedList implementation (from Week 3). Add the following sort functions to your LinkedList class. You will need to use while loops and a temporary Node\* to traverse the list:

* ascendingSelectionSort()
* descendingInsertionSort()
* ascendingBubbleSort()

Return to the tester file and add the code from above to add 100 random values to the DoublyLinkedList. Test each of your sort functions; call sort() and print().