Data Structures (810:052) Name:_____

1) Consider the linear/sequential search of an unsorted array for some target value. If the array contains n elements what would you expect the theta $\Theta(\ )$ notation to be for each of the following cases:

a) worst case

b) best case

c) average case

2) Consider the binary search of a sorted array for some target value. If the array contains n elements what would you expect the theta $\Theta(\ )$ notation to be for each of the following cases:

a) worst case

b) best case

c) average case

3) All *simple sort*s consist of two nested loops where:

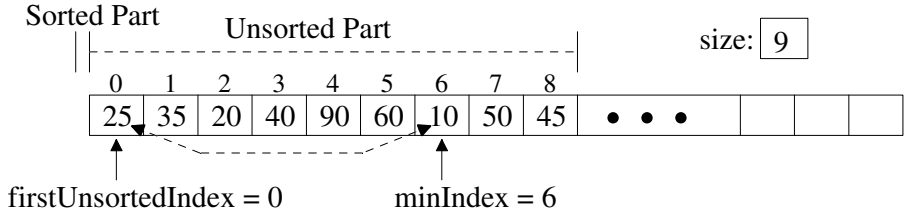
- the outer loop keeps track of the dividing line between the sorted and unsorted part with the sorted part growing by one in size each iteration of the outer loop.
    - the inner loop's job is to do the work to extend the sorted part's size by one.

Initially, the sorted part is typically empty. The simple sorts differ in how their inner loops perform their job.

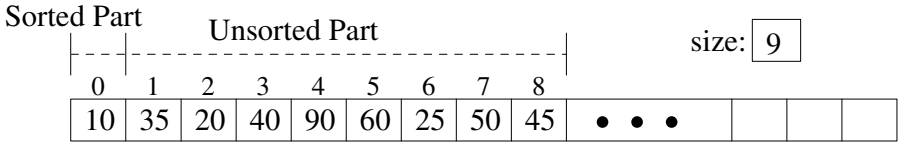
Selection sort is an example of a simple sort. Selection sort's inner loop scans the unsorted part of the array to find the minimum item. The minimum item in the unsorted part is then exchanged with the first unsorted item to extend the sorted part by one item.

At the start of the first iteration of the outer loop, the initial array is completely unsorted:

The inner loop scans the unsorted part and determines that the index of the minimum item, minIndex = 6.



After the inner loop (but still inside the outer loop), the item at minIndex is exchanged with the item at firstUnsortedIndex. Thus, extending the Sorted Part of the array by one item.



Consider the selection sort code below to find its worst-case $\Theta(\ )$ notation.

```
void selectionSort(int array[], int size) {
      int temp, testIndex, minIndex, firstUnsortedIndex;

      for (firstUnsortedIndex = 0; firstUnsortedIndex < size-1; firstUnsortedIndex++) {

            minIndex = firstUnsortedIndex;
            for (testIndex = firstUnsortedIndex+1; testIndex < size; testIndex++) {

                  if (array[testIndex] < array[minIndex]) {
                        minIndex = testIndex;
                  } // end if
            } // end for

            temp = array[firstUnsortedIndex];
            array[firstUnsortedIndex] = array[minIndex];
            array[minIndex] = temp;
      } // end for
} // end selectionSort
```

a) What statements are executed the most?

b) What is the worst-case number of element moves?

c) What is the worst-case number of element comparisons?

d) What is the overall worst-case $\Theta(\ )$ notation for selection sort?

Consider the bubble sort code below to find its worst-case $\Theta(\ )$ notation.

```
void bubbleSort(int array[], int size) {
    bool swap;
    int temp, test, lastUnsorted;

    lastUnsorted = size-1;
    do {
        swap = false;
        for (test = 0; test < lastUnsorted; test++) {
            if (array[test] > array[test + 1]) {
                temp = array[test];
                array[test] = array[test + 1];
                array[test + 1] = temp;
                swap = true;
            } // end if
        } // end for
        lastUnsorted--;
    } while (swap);
} // end bubbleSort
```

a) What statements are executed the most in the worst case?

b) What is the worst-case number of element moves?

c) What is the worst-case number of element comparisons?

d) What is the overall worst-case $\Theta(\ )$ notation for bubble sort?