Group 7

CSCI 313

HW1-- Question 3: Sorting 2D Arrays

The first function we implemented in this code was the TwoDtoOneD function. The function takes in a pointer to a pointer as well as an integer value for length. We then initialized a pointer called myarray that was of length "len." Finally, we used a for loop to transfer all of the values of the 2D array into a 1D array. Next, we created a swap function. This function took in the addresses of the two integer variables we wanted to swap, and using a temporary integer variable, switched the values.

With these two functions in place, we could begin coding our different sorts. First, we created the bubble sort. The first thing we do is multiply the dimensions of the 2D array to get the length of the 1D array. We do this because if we, for example, had a 2D array a[5][5], we could convert it to a 1D array a[25]. So, we first double the length of the 2D array, and call our TwoDtoOneD function to be able to sort this as a 1D function. We then use a for loop that compares each of the adjacent elements in the array. If the left value is greater than the right value, we call the swap function and swap the two values. This for loop continues until we have successfully ordered the array.

Next, we create the selection sort function. Similar to bubble sort, we find the new length of the 1D array by multiplying the dimensions of the 2D array and call the TwoDtoOneD function. Once we have our 1D array, we create an integer value to store the minimum index. Then we create a for loop that iterates through the entire array. We set the minimum index to 0 first and compare it to the value of the adjacent element. If the adjacent element is less than the

minimum index, we set the adjacent element equal to the minimum index and then swap the two values. We continue this process until we have iterated through the entire array.

Lastly, we create the insertion sort function. We begin by creating three integer variables. We have a key integer that we can continually refer to as well as an “i” and a “j” variable that are responsible for iterating through each element of the array (through for loops). The function will compare each of the adjacent elements. If the left value is greater than the right value, the right value will be shifted over and the left value will take its place.

Finally, our main function is responsible for producing a sample 2D array. It then prints the unsorted array as well as the sorted array.