Jean Gomez

Question 3

For each function I passed a double reference to the array and the number of rows and columns the array had. For the bubble sort I created a while loop that would exit only when the Boolean swapped variable was equal to false. Then I used two for loops that would compare the first two elements in the row and if the element to the left of it was greater than we would swap values and set the variable swapped equal to true. If we came to the end of that row, the final column, then we would check if there is another row under it. If this if statement was fulfilled, then we would compare the value in the last column of the current row to the first element in the row under it. If the value in the current row was greater than the value in the row below then we swap values and set the swapped equal to true. The reason for the swapped variable is so we can keep running the loop until all the values are in correct ascending order. When the swapped variable is set to false that means we have sorted all the values in the 2d array so there is no need to swap anymore.

For the insertion sort I passed in the same parameters as I did for the bubble sort. I also included a Boolean variable swapped that serves the same purpose as it did in the bubble sort. The double for loops were used to iterate through the 2d array with the outer loop being for rows and the inner loop being for columns. I created an integer key that was equal to value at index arr[0][1] this way I could compare to all the elements before it. The while loop inside the double for loops was used to check if the value before key was greater than it and if it was then we swap the index’s and if there were more values before the key then we will decrement the variable y to check if such values exist before the key. As long as the variable y was greater than or equal to zero it would run because that would mean we have reached the first value in that row. After we break out of the while loop then we set the next value in that row to key and repeat the process. If we had come to the end of the column then we would go to our if statement that checks if there is a row under it. Then if there was, we check if the key was greater than the element beneath and if so we swapped the index’s and set the variable swapped equal to true. Once we break out of the double for loops, we check our swapped variable and since it is equal to true, we repeat the process until the swapped variable comes back false letting us know that everything is in ascending order.

Lastly, for the selection sort is where I encountered the most difficulty because I kept backtracking when I was supposed to take the first element in the 2d array and compare it to all the other elements inside of it. Originally, I just wanted to use a while loop and double for loops, but it would still print out of order, so I had to switch methods. The first double for loops is so we can iterate through the whole 2d array. This just keeps track of where we are in the array. Then we create variables currentN that is meant to show us what index we are in. MinN was used so we can compare it to the other elements without making any changes to the actual index. Then minNRow and minNcol were created so that if we found an element that was less than the current index we would take its row and column position till we finished iterating through the whole array and if that element was the smallest then we swapped its position with the element we were comparing all others to. The first for loop inside the double for loops is used so we can compare all elements after the current element we are using in that row. My reason behind this was when I wanted to use only 2 for loops and would finish comparing currentN to all other elements I would come back to the original element I just swapped and it would have the elements out of order. So, this first for loop takes us to the position that is 1 column after the currentN and will compare all elements in that row to currentN without any backtracking issues. Then it’ll move on to the next for loop that’ll finish iterating through the rest of the element and find the position of the smallest element. After we finish comparing all elements to currentN we go on to the swap function that will swap the index we were comparing all other elements to with the smallest element position we saved with minNRow and minNcol.