**CSC175 Practice Assignment 6 Spring 2019 Name \_\_\_\_\_Donald Tvedt\_\_\_\_\_\_\_\_\_\_\_\_\_   
Directions:** Download this file and save as lastnamePracticeAssignment6SP19. Type all solutions on this document. Use equation editor when necessary. Upload Word document to Blackboard by **Saturday at 11:59 AM.** Let me know if you have any questions. **Show Work!**

1) Use the Euclidean Algorithm to find: (show each step of the algorithm)

a) GCD(540,395)

**540 = 1(395) + 145 = GCD(395, 145)**

**395 = 2(145) + 105 = GCD(145, 105)**

**145 = 1(105) + 40 = GCD(105, 40)**

**105 = 2(40) + 25 = GCD(40, 25)**

**40 = 1(25) + 15 = GCD(25, 15)**

**25 = 1(15) + 10 = GCD(15, 10)**

**15 = 1(10) + 5 = GCD(10, 5)**

**10 = 2(5) + 0**

**GCD(540,395) = 5**

b) GCD(675,56)

**675 = 12(56) + 3 = GCD(56, 3)**

**56 = 18(3) + 2 = GCD(3, 2)**

**3 = 1(2) + 1 = GCD(2,1)**

**2 = 2(1) + 0**

**GCD(675,56) = 1**

2) Consider the list 9 16 17 29 38 54 78

a) What is the most steps it could take to find a number if we traveled through the list one at a time? Why? **7 when you have to travel the list 1 at a time the most it would take would be the number of elements**

b) What is the most steps it could take to find a number of we traveled through the list using binary search? Why? **3 is most it would take to find a number using binary search, pick half, pick half again, then you have your number.**

c) Use binary search to find 38. Explain each step of the algorithm, showing the remaining list you are considering each time through the algorithm.

**9 16 17 29 38 54 78**

**Pick half way point = 29 so 38 is greater then 29 so take the 2nd half of the indexes.**

**38 54 78**

**Pick half way point = 54 so 38 is less then 54so take the 1st half of the indexes.**

**38**

**We have our answer = 38**

3) Give the Big O runtime for each algorithm. Explain your answer.

a) Begin Check\_Algorithm (List of numbers A)

Enter N (length of A)  
For i = 1 to N  
For k = i to N  
If A[i]<A[k]  
 print A[k]  
 End If  
End For  
End For

End Check\_Algorithm

**These arrays are nested and running through the same data so this is =**

b) Begin Check\_Again\_Algorithm (List of numbers A)

Enter N (length of A)  
For i = 2 to N  
If A[i]<A[1]  
 print A[i]  
 End If  
End For  
For k = 3 to N  
 If A[k]<A[2]  
 print A[k]  
 End If  
End For

End Check\_Again\_Algorithm

**Since these are separate for loops and we are not going through the entire array =**

c) Begin Summer\_Algorithm (List of numbers A where the first entry(input) is the month (output): 1=Jan, 2 = Feb, etc.)

Enter N (length of A)  
 If A[1]=6  
 Print “It is summer!”  
 Else If A[1]=7  
 Print “It is summer!”  
 Else If A[1]=8  
 Print “It is summer!”  
 Else  
 Print “It is not summer! ☹”  
 End If

End Summer\_Algorithm

**Since these are separate for loops and we are not going through the entire array =**

4) Explain what is happening for each of the following algorithms. Be as thorough as possible.

a) The Check\_Algorithm in 3a above

**We are nested looping 2 times through the array of numbers. We are then comparing the indexes in the Array and looking at if its less then if it is less then it will print the 2nd number.**

b) The Check\_Again\_Algorithm in 3b above

**We are looping through the array skipping the first index and comparing the 2nd through N index to the first index if its less then the first index we are printing the current index.**

**On the 2nd loop we are looping through the array skipping the first 2 indexes and comparing the 3rd through n indexes to 2nd index if its less then the 2nd then print the index.**

5) Consider the list 12 4 23 21 8 5

a) Explain how bubble sort works in your own words. Use bubble sort to sort the list. Show each swap.

**Bubble sort = walks an entire array and compares 2 indexes at a time when it finds that they are not in order it swaps those indexes and continues that until the end of the array. We are not done yet it will walk the entire array over and over until the entire array is sorted.**

**12 4 23 21 8 5**

**Pass#1 = (4 12) (21 23) (5 8)**

**Pass#2 = 4 12 21 (5 23) 8**

**Pass#3 = 4 12 (5 21) (8 23)**

**Pass#4 = 4 (5 12) (8 21) 23**

**Pass#4 = 4 5 (8 12) 21 23**

**Pass#5 = 4 5 8 12 21 23 = no swaps all sorted**

b) Explain how selection sort works in your own words. Use selection sort to sort the list. Show each swap.

**Selection Sort = Starting at the first index we look through the array for the smallest number and swaps the 2 indexes. Starting at the 2nd index we again look for the next smallest number and swap the numbers and rinse and repeat until completely sorted.**

**12 4 23 21 8 5**

**Pass#1 = (4) (12) 23 21 8 5**

**Pass#2 = 4 (5) 23 21 8 (12)**

**Pass#3 = 4 5 (8) 21 (23) 12**

**Pass#4 = 4 5 8 (12) 23 (21)**

**Pass#5 = 4 5 8 12 (21) (23)**

**Pass#5 = 4 5 8 12 21 23 = no swaps all sorted**

6) Answer each question.

a) What is the best case runtime for bubble sort? When does that occur?

**Best case is when the entire array is already sorted. You still need to iterate through the entire array.**

b) What is the worst case runtime for bubble sort? When does that occur?

**Worse case is when the entire array is in the wrong order. You need to interate through the entire array multiple times.**

c) Why is selection sort O(N2) for both best and worst case?

**Same as for the bubble sort you still need to iterate through the array multiple times to get the array sorted. I don’t see these sorts as being very efficient. But they do get the job done!**