**CSC175 Assignment 6 Spring 2019 Name \_\_\_\_\_\_\_\_\_\_Donald Tvedt\_\_\_\_\_\_\_\_\_\_\_\_\_   
Directions:** Download this file and save as lastnameAssignment6SP19. Type all solutions on this document. Use equation editor when necessary. Upload Word document to Blackboard by **Monday at 11:59 PM.**  
Points in [brackets]. Total: 60 points **Show work** and explain concepts thoroughly!  
[5] points for a professional looking document (directions, organization, neatness, etc.)  
  
1) [8] Use the Euclidean Algorithm to find: (show each step of the algorithm)

1. GDC(693,600)

**693 =1(600) + 93**

**600 = 6(93) + 42**

**93 = 2(42) + 9**

**42 = 4(9) + 6**

**9 = 1(6) + 3**

**6 = 2(3) + 0**

**GCD(693,600) = 3**

1. GCD(1386,880)

**1386= 1(880) + 506**

**880 = 1(506) + 374**

**506 = 1(374) + 132**

**374 = 2(132) + 110**

**132 = 1(110) + 22**

**110 = 5(22) + 0**

**GCD(1386,880) = 22**

2) [8] Consider the sorted list: 4 9 11 13 16 18 21 22 23 29 31

a) What is the most steps it could take to find a number if we traveled through the list one at a time? Why?

**Answer = 11, there are 11 indexes if you travel the entire list.**

1. What is the most steps it could take to find a number of we traveled through the list using binary search? Why? **Answer = 4, using the binary search taking half each time we will find the answer in 4 turns.**
2. Use binary search to find 9. Explain how you used the algorithm.

**Search for 9, half the number and check to see if greater, less than, or equal.**

**4 9 11 13 16 (18) 21 22 23 29 31 = 18 > 9 then take the bottom half**

**4 9 (11) 13 16 = 11 > 9 then take the bottom half**

**(4) 9 = 4 < 9 then take the upper half**

**9 = 9=9 we have solved the search in 4 steps**

3) a) [2] Give the Big O runtime for the following algorithm. Explain your answer.

Begin MathAlgorithm (List of numbers A)

Enter N (length of A)  
print N “ is the length of A.”  
M=N\*3  
print M “ is a multiple of 3.”  
If N<5 then  
 print “A has less than 5 elements.”  
Else if N>9  
 print “A has more than 9 elements.”  
Else  
 print “A has between 5 and 9 elements inclusively.”  
W = A[N]\*5  
print W “ is a multiple of 5.”

End MathAlgorithm

**We are only doing multiple things on 1 element, there this is a Big O constant.**

b) [4] For the MathAlgorithm above, what would the output be if A = (32, 54, 21, 16, 39, 123, 14, 27)?

**8 is the length of A.**

**24 is a multiple of 3.**

**A has between 5 and 9 elements inclusively.**

1. **a multiple of 5.**

c) [2] Give the Big O runtime for the following algorithm. Explain your answer.

Begin Find17Algorithm (List of unsorted numbers A which includes the element 17)

Enter N (length of A)  
Conduct Quick Sort on A  
Conduct Binary Search on A to find element 17

End Find17Algorithm

**= Even those we are doing multiple things the worst case is the Quick Sort therefore it the worst case scenario.**

d)[2] How would the Big O runtime for the Find17Algorithm above change if A was sorted?

**A sorted list would shorten it down to the Binary Search as the worst case scenario.**

e) [2] Give the Big O runtime for the following algorithm. Explain your answer.

Begin InterestingAlgorithm (List of numbers A)

Enter N (length of A)  
For i = 1 to N-1  
For j = i+1 to N  
If A[i]=A[j]  
 print A[i]  
End If  
End For  
End For  
For k = 1 to N  
If A[k]<1 or A[k]>99  
 print A[k] “ is not a valid number.”  
Else  
 print A[k] “ is a valid number.”  
End If  
End For

End InterestingAlgorithm

**We are doing nested looping through A**

f) [2] What does the InterestingAlgorithm do? Be as clear as possible.

**The first section of code with the nested for loops would only print the number if there we’re duplicate numbers entered consecutively.**

**The second section will print as many times as there are indexes in the array. Your printed text will alternate between 2 lines and they will be dependent on if the number is less then 1 or greater then 99 you will get “the number is not a valid number.” If the number is between 1 and 99 then you will get “the number is a valid number.”**

4) Consider the list 21 11 23 16 13 22 18

a) [5] Use bubble sort to sort the list. Show each swap. How many swaps took place?

**21 11 23 16 13 22 18 = Start sort**

**(11 21) 23 16 13 22 18 = 1 swap**

**11 21 (16 23) 13 22 18 = 2 swap**

**11 (16 21) 23 13 22 18 = 3 swap**

**11 16 21 (13 23) 22 18 = 4 swap**

**11 16 (13 21) 23 22 18 = 5 swap**

**11 (13 16 ) 21 23 22 18 = 6 swap**

**11 13 16 21 (22 23) 18 = 7 swap**

**11 13 16 21 22 (18 23) = 8 swap**

**11 13 16 21 (18 22) 23 = 9 swap**

**11 13 16 (18 21) 22 23 = 10 swaps = Done!**

b) [7] Explain how merge sort works in your own words. Use merge sort to sort the list. Show each step.

**Split the data in half over and over until everything is in singles. After all in singles then merge into sorted pairs, when possible. Then merge into sorted groups of 4 then 8 then 16, etc. until everything is sorted.**

**21 11 23 16 13 22 18 = Start sort**

**21 11 23 16 | 13 22 18 = split**

**21 11 | 23 16 | 13 22 | 18 = split**

**21 | 11 | 23 | 16 | 13 | 22 | 18 = split everything in singles**

**(11 21) | (16 23) | (13 22) | 18 = merge into sorted pairs**

**(11 16 21 23) | (13 22) | 18 = merge into sorted 4**

**(11 13 16 18 21 22 23) = merge into sorted 8**

c) [7] Explain how insertion sort works in your own words. Use insertion sort to sort the list. Show each step.

Starting at the left you are comparing the digit to the left (first position nothing to do) to see if its in order, if not then swap until it get into sorted order.

**21 11 23 16 13 22 18 = Start sort start at 21 nothing to do move to 11**

**(11 21) 23 16 13 22 18 = 11 not sorted so we swap it into the correct position and move to 23**

**11 21 23 16 13 22 18 = 23 is sorted so we move to 16**

**11 (16) (21) (23) 13 22 18 = 16 not sorted so we swap it to the correct position and move to 13**

**11 (13) (16) (21) (23) 22 18 = 13 not sorted so we swap it to the correct position and move to 22**

**11 13 16 21 (22) (23) 18 = 22 not sorted so we swap it to the correct position and move to 18**

**11 13 16 (18) (21) (22) (23) = 18 not sorted so we swap it to the correct position we finished**

5) [6] Answer each question.

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

**When the array is already sorted is when we get the best run time.**

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

**When the array is in its opposite order and every index needs to be swapped.**

c)What is Big O notation for merge sort? Generally does it differ from best to worst case runtime? Why do you think that is the case?

**Big O notation for Merge sort. You follow the same pattern no matter if its already sorted or not sorted. Therefore all the cases are the same.**