**HW1 CSCD320**

**Question 1, programing is NOT required.**

**1.**  Read the following code and answer questions.

**public** **static** **void** one(**int** a[], **int** j, **int** m)

{

**int** i, temp;

**if**(j < m) {

**for**( i = j; i <=m; i ++ ){

**if**( a[i] < a[j] ) {

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}//end of for

j++;

*one*(a, j, m);

}//end of outer if

}//end of one method

**public** **static** **void** main(String args[]) {

**int** a[] = {2, 5, 1, 7, 9, 3, 6, 8};

*one*(a, 0, a.length - 1);

System.*out*.println(Arrays.*toString*(a));

}

(a)Summarize what task(s) the method ***one*** perform? What will the **main()** method print on the screen after execution? ( 10 points)

(b) Determine the Growth Rate Function(GRF) for the method ***one***, please justify how did you get your GRF. As a separate task, then, you show the time complexity using Big-Oh notation. (40 points)

**Question 2. Programming is required.**

# Please design an algorithm quickSearch() with the worst time complexity of log(n). (50 points)

# a) *The method int quickSearch(int array[], int value)* will search an sorted input array and return the index of an array element X, such that X is greater than or equal to the value to be searched. In other words, if value is not found in the input array, the algorithm returns the index of the closest item in array (but greater). For example, with an input array A = [ 1, 3, 5, 7, 9, 14, 16, 19 ],

# if we perform quickSearch(A, 8), it return 4, the index of number 9.

# if we perform quickSearch(A, 19), it return 7, the index of number 19.

# if we perform quickSearch(A, 20), it return -1, indicating there is no such value that is bigger than or equal to 20 in array A.

# if we perform quickSearch(A, 6), it return 3, the index of number 7.

# if we perform quickSearch(A, -1), it return 0, the index of number 1 in array A.

**Please write a single java source file Tester.java that contains the implementation of quickSearch() and a main() method in which you call the quickSearch() method. In your main(), please carry out the test cases listed in the blow. Your java program has to display the following information on the standard output.**

# *Using an input array A = [ 1, 3, 5, 7, 9, 14, 16, 19 ],*

# *performing quickSearch(A, 8), it return 4, the index of number 9.*

# *performing quickSearch(A, 19), it return 7, the index of number 19.*

# *performing quickSearch(A, 20), it return -1, indicating there is no such value that is bigger than or equal to 20 in array A.*

# *performing quickSearch(A, 6), it return 3, the index of number 7.*

# *performing quickSearch(A, -1), it return 0, the index of number 1 in array..*

b)Please analyze whether the time complexity of your algorithm is (log n). **If you cannot provide a *O(log n)* algorithm, you lose a lot of points. Hint: THIS IS A MODIFIED BINARY SEARCH.**

c) You can have your own design for any details that **have not** specified in this document.

**To Turn in,**

1. **Please write the questions and your answers for 1a, 1b, and 2b in a single PDF file.**
2. **Please wrap up your PDF file and your Tester.java file into a single zip file, named with your lastname + initial of first name + hw1.zip. For example, smithjhw1.zip for John Smith.**
3. **Turn in the single zip file on Canvas for HW1.**