# Data Structures and Algorithms

# INFO 6205

# Homework 5

# Due: October 11, 2019

1. Consider the following, Input Data: {7, 38, 3, 9, 82, 10, 31, 24}

a) Graphically build a Circular queue for input data. Discuss and show Head

and Tail pointers at each step:

i) enqueue all input data

ii) dequeue three elements

iii) enqueue two elements

iv) dequeue all elements

b) Write Java code for the Circular queue, provide enqueue, dequeue,

isEmpty, isFull, and displayQueue methods, to show the status of the queue

with steps described in (a). Compile code and Run with input data.

2. Consider signed byte X, and unsigned byte Y. What are the possible values for both

X and Y can have?

3. Consider the following example for QuickSort:

<http://interactivepython.org/courselib/static/pythonds/SortSearch/TheQuickSort.html>

a) Complete the example, b) Compile the code, and run it with Input data example

4. Consider the following MergeSort algorithm with Input Data:

{27, 43, 38, 3, 9, 82, 10}

MergeSort(arr[], l, r)

If r > l

1.Find the middle point to divide the array into

two halves: middle m = (l+r)/2

2.Call mergeSort for first half:

Call mergeSort(arr, l, m)

3. Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

4.Merge the two halves sorted in step 2 and 3:

Call merge(arr, l, m, r)

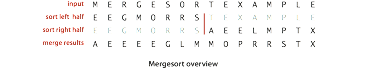
a) Sort the data Graphically, show step-by-step, recursion on Stack

and what is Termination point?

b) Write Java code and Compile and Run with provided data.

5. Consider attached image Boston.jpg. Write a program to sort the image Pixels by “brightness”. You program for three sorting algorithms: HealSort, QuickSort, and MergeSort. You need to sort the Pixel array size of the image in descending order and show the runtime time complexity of each Sorting algorithm and compare.

NOTES: You may NOT use any Java library function for sorting. You should use ONLY the Sorting Java code I provided in class. The Pixel sorting should start from (0,0) to (high,high) for Brightness. For each Pixel, you need to convert RGB color to appropriate intensity. Use intensity formula:



I = 0.2989R + 0.5870G + 0.1140B. If the current pixel Intensity is larger than the next pixel intensity, you need to swap, going in descending order.

You may need the following classes:

Java.awt.image.BufferedImage: image class.

eg: image = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_ARGB);

java.util.\*: collection of List data types.

javax.imageio.ImageIO: for reading/writing images to file

6. Consider mergeSort algorithm for the following input string. Show the stack operations push and pop step by step for call mergeSort(arr, l, m) and call mergeSort(arr, m+1, r). Note: I don’t need the entire program, just show step by step stack push and pop operations, the recursive Tree structure