# Data Structures and Algorithms

# INFO 6205

# Homework 4

# Due: February 10, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework4.zip and submit it via the Drop Box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. Java String hashcode is the following: pasted-image.tiff

What is the hashCode 32-bit integer number for string =“Hello Students”?

1. Mathematically by hand,

**Ans:** s = “Hello Students”

s.hashcode() = s(0)\*31^(13) + s(1)\*31^(12) + s(2)\*31^(11) + s(3)\*31^(10) + 0 + s(5)\*31^(8) + s(6)\*31^(7) + s(7)\*31^(6) + s(8)\*31^(5) + s(9)\*31^(4) + s(10)\*31^(3) + s(11)\*31^(2) + s(12)\*31^(1) + s(13)\*31^(0)

s.hashcode() = 72\*31^(13) + 101\*31^(12) + 108\*31^(11) + 108\*31^(10) + 111\*31^(9) + 0 + 115\*31^(7)+116\*31^(6)+ 117\*31^(5)+ 100\*31^(4)+ 101\*31^(3)+ 110\*31^(2)+ 116\*31^(1)+ 115\*31^(0)

s.hashcode() = 1.840452848006898E21 ~ **-1752069786**

B) Write Java code

2. Consider the following code for User class.

A) Discuss code in details

B) Write Java code to test User class with multiple test cases to test equals,

hashCode and CompareTo methods.

public class User implements Comparable<User> {

private String name;

private int id;

private Date birth;

public User (String name, int id, Date birth)

{ this.name = name; this.id = id; this.birth = birth; }

@Override

public boolean equals(Object other) {

if (this == other) return true;

if (other == null || (this.getClass() != other.getClass()))

{ return false; }

User guest = (User) other;

return(this.id == guest.id) &&

(this.name = null && name.equals(guest.name)) &&

(this.dob != null && dob.equals(guest.birth));

}

@Override

public int hashCode() {

int result = 0;

result = 31\*result + id;

result = 31\*result + (name !=null ? name.hashCode() : 0);

result = 31\*result + (birth !=null ? dob.hashCode() : 0);

return result;

}

@Override

public int compareTo(User o) {

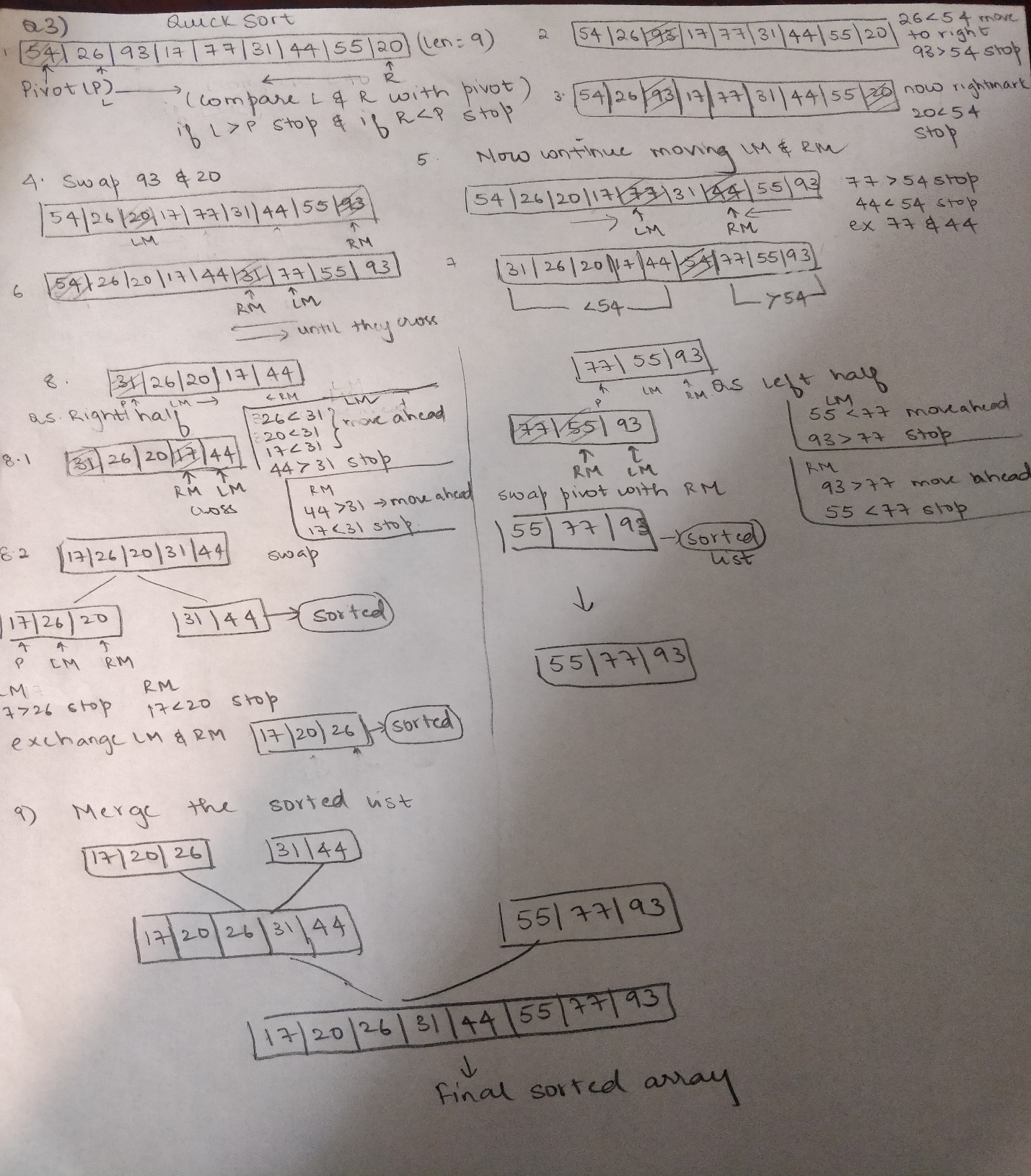
return this.id - o.id; }

}

}

3. Consider the following example discussed in class for QuickSort, Complete the example

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



4. Consider megeSort algorithm for array {38, 10, 43, 3, 9, 82, 27}. 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.

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:

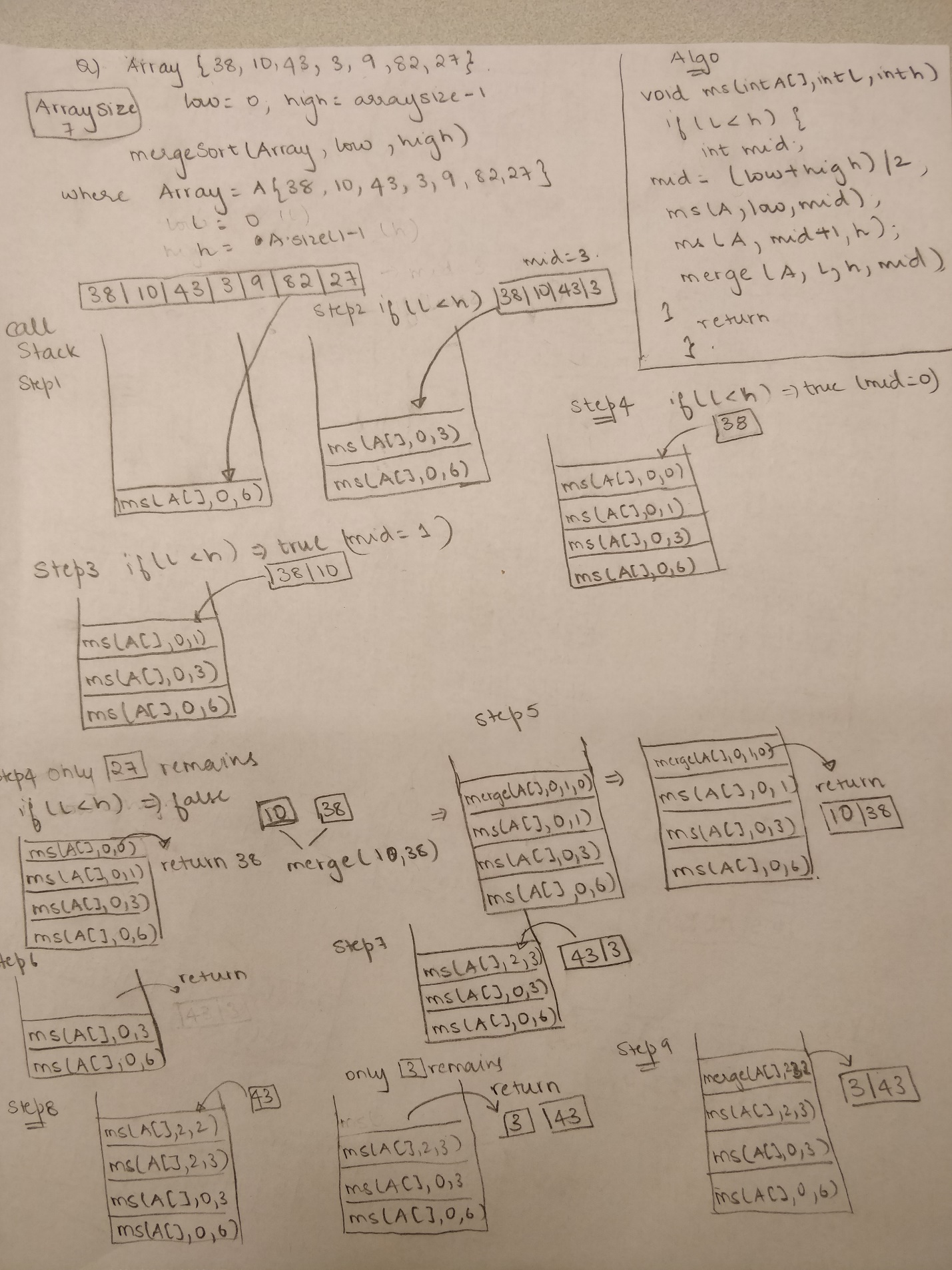
Call mergeSort(arr, l, m)

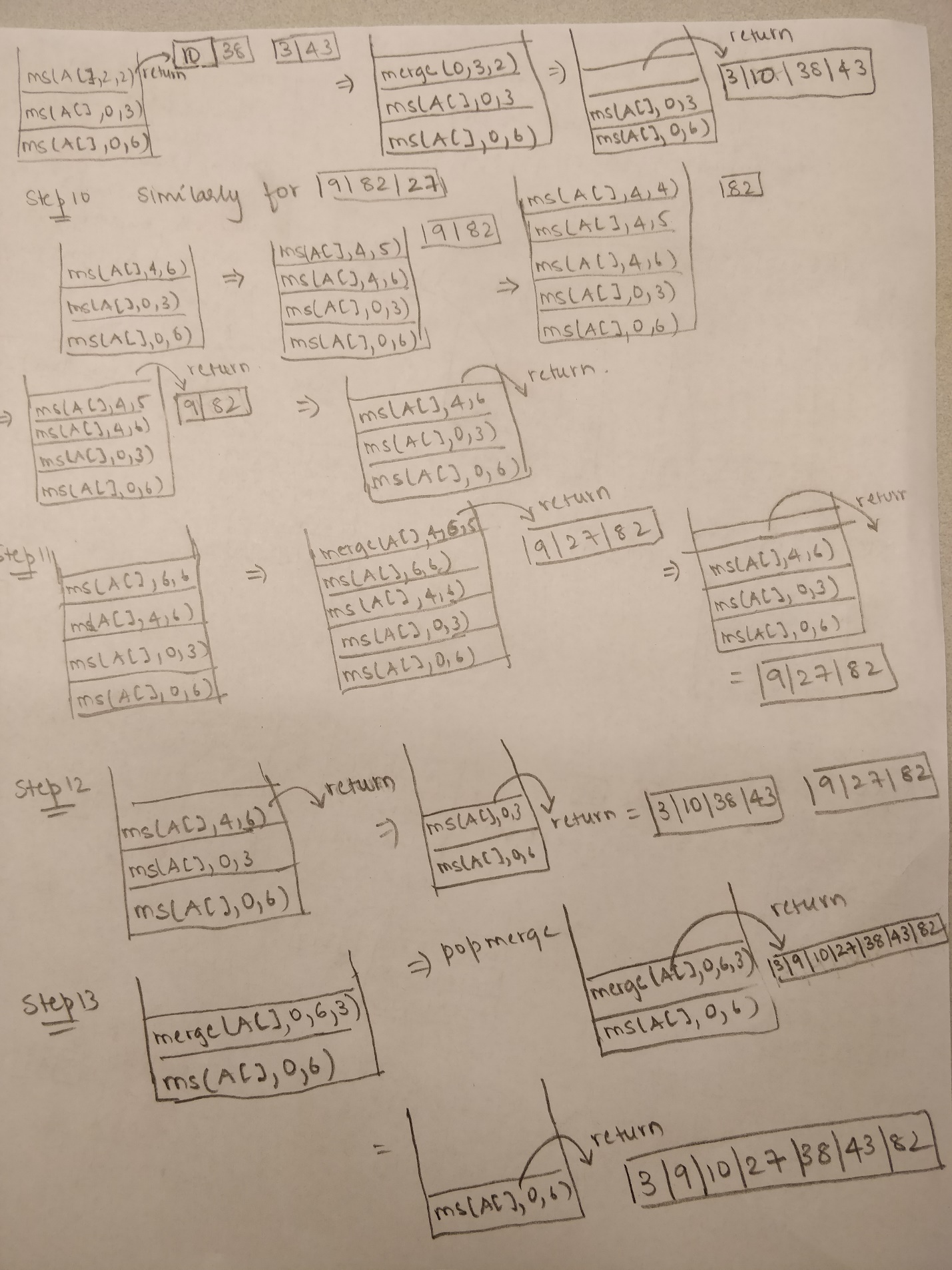
3. Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

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

Call merge(arr, l, m, r)





5. Consider attached image Boston.jpg. Write a program to sort the image Pixels by “brightness”. You program for four sorting algorithms: InsertionSort, HealSort, QuickSort, TimSort, 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