**PART II:12/52 points**

Name: Nicholas Passantion

1. [12 points] Write a method in Java that returns the total number of trailing zeroes for all integers from 1 to its parameter n; given 5, it returns 0 + 1 + 0 + 2 + 0 = 3. The trailing zeroes of an integer a are the zeroes following the last 1 in a's binary representation (ex: 0100 ->2; 0101->0). Using big-O notation in terms of its parameter n and provide the time complexity of your algorithm.

[4 points]

* The program failed for the test case where n is 8. The trailing zeroes should be 7, but the program produced 8 as an output
* It is very hard to read and maintain the code. I would suggest you to re-design your solution as described in the problem.
* Algorithm analysis: is it n2 or nlogn? You have n = n-pow, but pow is not actually one. You are skipping some n values.
* Also, I would like to let you know that you can design an algorithm with O(logn) complexity
* Program needs documentation

1. [12 points] Write a program that, given two sorted arrays of n in values, prints all elements that appear in both arrays, in sorted order. The running time of your program should be proportional to n in the worst case (Exercise 1.4.12)

[0 points] Arrays are already sorted, why do you want to sort them again with binary search tree. As given in the problem, the running time of your program should be proportional to n in the worst canse.

Also, I think there is a misunderstanding in this question. You are asked to find the intersection between two sorted arrays, elements that appear in both arrays.

1. [12 points] Write a program that, given an array a[] of n distinct integers, finds a strict local minimum: an entry a[i] that is strictly less than its neighbors. Each internal entry (other than a[0] and a[n-1] ) has 2 neighbors. Your program should use ~2(lgn) compares in the worst case (Exercise 1.4.18). Provide tests for all possible cases.

[0 points] The program fails for the input array {1,4,5,6}. Also, it doesn’t have O(logn) running time complexity. You are asked to design a solution with running time of O(logn).

1. [12 points] Design and implement an algorithm in Java that performs substring pattern matching. Your algorithm needs to determine if the text contains the given pattern and returns position of the pattern.

Input: A text string t and a pattern string p.

Output: Does t contain the pattern p as a substring, and if so where?

What is the complexity of your algorithm. Please show your work.

[4 points]

* The program fails for the case match("This is a test", "tell");
* Algorithm analysis part is missing. Since there are two strings with different lengths (say n and m), the running time complexity should be in form of n and m.

1. [4 points] Provide a sample program that draws points, lines, squares and circles using StdDraw class provided by the textbook. You can learn more about this class at

<https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/StdDraw.html>.

[4 points]