Assignment 3

**Section 1: Pen-and-paper Exercises**

1. Let L be a list of numbers in non-decreasing order, and x be a given number. Describe an algorithm that counts the number of elements in L whose values are x (English description 5 points + Pseudocode 5 points). For example, if L = {1.3, 2.1, 2.1, 2.1, 2.1, 6.7, 7.5, 7.5, 8.6, 9.0} and x = 2.1 then the output of your algorithm should be 4. **Your algorithm should run in O(log n) time** (10 points).

**Important: In all of the assignments of this course, when you are asked to give an**

**algorithm for a problem, you are (unless otherwise indicated) expected to**

1. **describe the idea behind your algorithm in English (2 points);**

Start a loop and check to see if the middle is x, greater than x, or less than x. If the middle is equal to x, then run through two loops. One is from the start ot the middle and the other is the middle to the end and increment the counter when a number has multiples. If mid was greater or less than x then respectively add or subtract one to the middle and run through the loop again. Eventually, the loop will run for when x is equal to mid and when the inner loops are completed the flag will flip to false making the outer loop end. Then the count is returned for the values.

1. **provide pseudocode (5 points);**

loop until flag is false

is mid = x

loop and increment the counter for each repetition of a number

if mid is > x then set end to middle-1

if mid is < x then set beg to middle+1

return count

1. **analyze its running time (3 points).**

The running time is O(log N)

**Regarding requirement (iii): Unless otherwise specified, show the steps of your analysis and present your result using big-O.**

**Note: This problem will be discussed in class. Algorithms that are O(n) or slower will be scored out of 3 points.**

1. You have n coins (n may be even or odd) such that n-1 coins are of the same weight and one coin is heavier than the other coins.

You have a balance scale: you can put any number of coins on each side of the scale at one time, and it will tell you if the two sides weigh the same, or which side is lighter if they do not weigh the same.

Outline an algorithm for finding the coin with different weight.

The number of weighing’s using your algorithm should be O(log n).

**Full credit (8 points) will be awarded for an algorithm that is O(log n). Algorithms that are O(n) or slower will be scored out of 3 points.**

1. **describe the idea behind your algorithm in English (5 points);**

Use a binary search to find out which coin is the different weight. Put half of the coins on each side of the scale and see which side is heavier. Then take all of the coins off the lighter side and split the left-over coins to the two sides and repeat. Again, one of the sides will weigh less than the others so you can remove half again. This is reduction and recursion.

1. **analyze its running time (3 points).**

O(Log n)

**For this problem, you do NOT need to write the pseudocode.**

1. What is the running time of Binary Search if we use it to search for a number in a sorted linked list?

**(3 points) your answer.**

O (n)

**(5 points) justification for your answer.**

This is because it will be proportional to the number of items in the linked list.

1. Given a positive integer x, find square root of it. If x is not a perfect square, then return floor (round down).

Examples:

Input: x = 4

Output: 2

Input: x = 11

Output: 3

Outline an algorithm for finding square root of x. Expected in O(log n) time.

**Full credit (10 points) will be awarded for an algorithm that is O(log n). Algorithms that are O(n) or slower will be scored out of 3 points.**

**Note: You should NOT use existing functions like math.sqrt() to obtain the square root of x. Create your own function. Solutions that use existing functions will receive 0 points.**

**(i) describe the idea behind your algorithm in English (2 points);**

Start with 0 and end with x. if x is equal to 0 or 1 then return x as the answer. While the start value is greater than or equal to the end value (x) do this: find middle by adding the start and end then dividing by 2, compare middle squared to x and if they are equal then return the middle since x is a perfect square. If x is greater than middle squared do a binary search between the middle+1 and the end, if x is less than middle squared do a binary search between the start and the middle. At the end, return the answer.

**(ii) provide pseudocode (5 points);**

Start is 1, end is x

While start <= end

Middle is (start = end)/2

If middle squared is x, return middle //perfect square

If middle squared is < x, start is middle + 1 and answer is middle

Else end=middle – 1

Return the answer

**(iii) analyze its running time (3 points).**

it would be O(log n) since it isn’t exponential but rather halving each iteration. (n 🡪 n/2 🡪 n/4 🡪 n/8 🡪 n/16 and so on) We also know that Binary search is O(log n) time complexity.

**Section 2: Java Implementation**

1. Implement problem 1 in Java (30 points).

Note:

Find a file called Problem1.java in assignment 3 folder.

Complete the method of count().

Test your method in the main method provided.

**Programs that are O(n) or slower will be scored out of 5 points.**

1. Implement problem 4 in Java (30 points).

Note:

Find a file called Problem4.java in assignment 3 folder.

Complete the method of squareroot().

Test your method in the main method provided.

**Programs that are O(n) or slower will be scored out of 5 points.**

**Programs that use existing functions like math.sqrt() will receive 0 points.**

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_3'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**