Homework 02 (Due: Wednesday, February 26, 2020)

CSCE 310

Instructions

This assignment consists of 6 analytical problems and 2 programming problems. Your solutions to the analytical problems must be submitted, as one PDF, via webhandin. While handwritten (then scanned) solutions to the analytical problems are acceptable, you are strongly encouraged to typeset your solutions in LaTeX or a word processor with an equation editor. The legibility of your solutions is of great importance. **It is required that your PDF's filename not include spaces, percent signs, pound symbols, or parentheses.**

Programming Assignment

Your methods will be tested on the `cse.unl.edu` server, using `gcc version 4.8.5 (SUSE Linux))`. To ensure proper execution, you should test your submission in the webgrader

You will submit `csce310homework02part01.h`, `csce310homework02part02.h`, `csce310homework02part01.cpp`, `csce310homework02part02.cpp` (and maybe `csce310homework02part03.h` and `csce310homework02part03.cpp`), along with your PDF, via webhandin. Starter code can be found in Canvas, see the announcement.

`overlappingIntervals`

`overlappingIntervals` is an adaptation of Exercise 6.1.8 on page 207. `overlappingIntervals` is a function that should take two (2) vectors of n double values each as input and return the maximum number of open intervals that overlap at a single point. The first vector defines the starting point of each interval and the second vector defines the ending point of each interval. It may be assumed that the first vector will be in ascending order.

`sumToN`

`sumToN` is an adaptation of Exercise 6.1.7 on page 206. `sumToN` will take two arguments: a vector of double values and another double value. The function will return `true` if two unique values in the array sum to the quantitiy of the second input value. It may be assumed that the vector will be in ascending order.

`averageComparisions` (10 Points Extra Credit or Honors Contract)

Given an array of n integers, return the average number of comparisons that would be required to successfully find an element in the array using binary search. You may assume that the values in the array will be provided in ascending order. When more than one element can be chosen in the search, choose the element with a smaller value.

General Guidelines

Sample header, source, and testing files have been provided. You may modify the `.h` and `.cpp` files as needed, but you will only be turning in the four/six files mentioned above. The webgrader will be compiling the code with the command `g++ -o /path/to/executable.out /path/to/source/files/*.cpp` for each part, but I will only be copying the files I asked for out of your submission and into separate directories for Part 1, Part 2, and Part 3.

Written Assignment

Question 1 (10 points)
Question 5.3.2 in *The Design and Analysis of Algorithms*

Question 2 (10 points)

(a) Draw a binary tree with 10 nodes labeled $0, 1, \ldots, 9$ in such a way that the inorder and preorder traversals of the tree yield the following lists: $8, 2, 3, 5, 6, 4, 0, 9, 1, 7$ (inorder) and $0, 5, 8, 3, 2, 6, 4, 1, 9, 7$ (preorder).

(b) Give an example of two permutations of the same n labels $0, 1, \ldots, n-1$ that cannot be inorder and postorder traversal lists of the same binary tree.

(c) Design an algorithm that constructs a binary tree for which two given lists of n labels $0, 1, \ldots, n-1$ are generated by the inorder and postorder traversals of the tree. Your algorithm should also indentify inputs for which the problem has no solution.

Question 3 (10 points)
Estimate how many searches will be needed to justify time spent on presorting an array of 10^1 elements if sorting is done by mergesort and searching is done by binary search. You may assume that all searches are for elements known to be in the array. What about an array of 10^5 elements?

Question 4 (10 points)
Question 6.1.1 in *The Design and Analysis of Algorithms*

Question 5 (10 points)
Question 6.1.9 in *The Design and Analysis of Algorithms*

Question 6 (10 points)
Question 6.5.1 in *The Design and Analysis of Algorithms*

webgrader Notes

The webgrader should take roughly 60 seconds to complete running. Evidence of successfully running the webgrader consists of producing a PDF successfully in the directory `https://cse.unl.edu/~cse310/reports/02/USER/USER_02.pdf`, where `USER` is your CSE username.

Your submission's success against each test case will be determined by use of `diff`. Your output must be exact. If you are intending to use debugging output, send that output to stderr, not stdout. Only send output you intend to be matched against the solution output to stdout.

Point Allocation

Question	Points
Question 1	10
Question 2	10
Question 3	10
Question 4	10
Question 5	10
Question 6	10
`overlappingIntervals`	
Test Cases	1×15
Compilation	5
`overlappingIntervals` Total	20
`sumToN`	
Test Cases	1×15
Compilation	5
`sumToN` Total	20
Total	100