Assignment 2

**Date Assigned: 09/22/2017**

**Due: Midnight 10/05/2017 on iLearn**

**Please read turn-in checklist at the end of this document before you start doing exercises.**

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

1. Consider the **interval SCHEDULING problem (NOT the interval PARTITIONING problem)** discussed in class. Design a greedy algorithm to solve the problem, and prove that the greedy solution is the optimal solution.

**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**

**(i) describe the idea behind your algorithm in English (10 points);**

**(ii) provide pseudocode (10 points);**

**(iii) analyze its running time (10 points).**

**Regarding requirement (iii): Unless otherwise specified, show the steps of your analysis**

**and present your result using big-O.**

**For this particular problem, we ask you for a proof (10 points).**

1. Consider the segment covering problem discussed in class. Design a greedy algorithm to solve the problem, and prove that the greedy solution is the optimal solution.

**Section 2: Java Implementation**

1. Implement the **interval SCHEDULING problem (NOT the interval PARTITIONING problem)** in Java.

Note:

Find a file called IntervalScheduling.java in assignment 2 folder.

Complete the method of greedyscheduling().

Test your method in the main method provided following the comments.

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

**algorithm for a problem, your code will be evaluated based on:**

**5 points - Execution**

**Each file must run without error or warning on valid input described in the main method provided.**

**5 points - Within Code Documentation**

**Is the code documented for obvious understanding of the use, preconditions, and postconditions of each function?**

**20 points - Correctness**

**Is the algorithm implemented correctly? Does your method pass the test?**

1. Implement the **interval PARTITIONING problem (NOT the interval SCHEDULNING problem)** in Java.

Note:

Find a file called Intervalpartitioning.java in assignment 2 folder.

Complete the method of greedypartitioning().

Test your method in the main method provided following the comments.

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

**algorithm for a problem, your code will be evaluated based on:**

**5 points - Execution**

**Each file must run without error or warning on valid input described in the main method provided.**

**5 points - Within Code Documentation**

**Is the code documented for obvious understanding of the use, preconditions, and postconditions of each function?**

**20 points - Correctness**

**Is the algorithm implemented correctly? Does your method pass the test?**

1. Implement the Coin Changing Problem in Java.

Note:

Find a file called Coinchange.java in assignment 2 folder.

Complete the method of greedycoinchange().

Test your method in the main method provided following the comments.

1. Implement the Fractional Knapsack Problem in Java.

Note:

Find a file called Fractionalknapsack.java in assignment 2 folder.

Complete the method of greedyfractionalknapsack().

Test your method in the main method provided following the comments

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), 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\_2'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**