COMP 1406Z – Fall 2023
Introduction to Computer Science II
Final Exam – Friday, December 22nd 2023

Exam Details

The exam will be submitted through Brightspace. The exam materials contained in the 1406-exam-materials.zip include an IntelliJ code project and a text file for the short answer problem. There are three problems in total and the exam will run from 9am-12pm.

To submit your exam, create a single .zip file with all your exam materials (IntelliJ project and short answer file) and submit it to the '1406 Exam' assignment submission on Brightspace. The official end time for the exam is 12pm (Eastern Time) but you will have until 12:15pm (Eastern Time) to submit your exam. **Do not wait until the last minute to submit the exam. Brightspace's clock may not be exactly synchronized with your own and no late exam submissions will be accepted**. After submitting and before the submission cut-off, you should download your own zip file, extract its contents, and ensure the files have the correct content.

You are not allowed to consult other students, TAs, or anybody else while completing the exam. You may use any IDE or text editor you generally use to write your code but must submit a valid IntelliJ project. You can execute your programs as many times as you want to debug and fix your code. The BinaryTreeTestProgram class and the StringCollectionTester class have been included in the IntelliJ project to provide some basic tests for your implementations. You may use any standard Java code. The solutions you create will be evaluated based on correctness, efficiency, and the overall code quality (e.g., OOP quality, code reuse, generalizability, etc.).

Problem 1 (15 marks)

Write your answers for this problem in the shortanswer.txt file. Create a class hierarchy for the following classes to be used in a software system modelling a university: Person, Exam, Employee, Instructor, Student, FullTimeStudent, Midterm, Professor, PartTimeStudent, CourseWork, Assignment, ClassRoom. For each class, indicate what other class it would inherit from, if any (e.g., "Student extends Exam"), and include a brief explanation of why you think this class should (or should not) inherit from another. Note that you do not need to fully develop the classes. You only need to indicate which classes extend from others and provide your explanation. Your

explanation could refer to some state or behaviour that you anticipate the classes having. You may add additional classes into the design if you think it is appropriate (provide justification). If you are unfamiliar with what one of the class names represents, write your assumptions about what that class would represent within the system and incorporate it into the class hierarchy based on those assumptions. Within reason, correctly reasoning about the class hierarchy can be considered just as important as recreating the expected solution.

Problem 2 (25 marks)

The code for this part can be found in the 1406-Exam-Code IntelliJ project included within the exam zip. If you do not use IntelliJ, you can copy the Java files from the src directory into whatever IDE you regularly use. Ensure you copy the files back into IntelliJ and submit a valid IntelliJ project with your exam materials.

For this problem, you are required to write an implementation of the provided StringCollection interface. A StringCollection will store Strings in a way similar to a list (e.g., maintaining order, allowing duplicates). A StringCollection, however, has some additional functionality that is not provided by a regular list implementation, like the getFrequency() and getUniqueStringCount() methods. Each method defined in the interface has an associated comment that describes how the method should work.

Write your implementation in the StringCollectionImplementation.java file. You are free to store the data within your class in any way, including using Java's collection classes. You should provide a comment in your code justifying your choice(s) for data storage. You should focus on decreasing the runtime complexity of your solution. Several methods contain a hint as to what your target runtime complexity should be. You can create additional classes if you have a justification to do so (note your reasoning in a comment within any created class). You should make an effort to reduce duplicated code and ensure that your code is written in a quality way (e.g., error free, generalized, good variable names, etc.). **You may not add to or change the StringCollection interface** but may add additional methods into your implementation class if desired. The provided StringCollectionTester class can be used to run some basic tests on your implementation.

Problem 3 (10 marks)

The code for this part can be found in the 1406-Exam-Code IntelliJ project included within the exam zip. If you do not use IntelliJ, you can copy the Java files from the src directory into whatever IDE you regularly use. Ensure you copy the files back into IntelliJ and submit a valid IntelliJ project with your exam materials.

The BinaryTree class within the exam IntelliJ project contains an implementation of a binary tree that does not use the sentinel approach covered in the lectures/readings (i.e., child nodes may be null). You must implement the boolean sameAs(BinaryTree otherRoot) method within this class. This method must return true if the calling BinaryTree is the same as the argument BinaryTree and false otherwise. Two binary trees should be considered the same if they have the same node structure (i.e., exact same shape) and each matching node has the same value. In other words, the data stored in the two trees is identical. The provided BinaryTreeTestProgram can be used to run some basic tests on your implementation.