Review for FPP Final
Final Draft

The final exam will be a paper exam, without notes or access to a computer.

1. **T/F, Multiple Choice, Short Answer.** These will draw upon the following list of topics from Lessons 7-12.

    a. *Recursion.* Be familiar with the following points:
        - What has to be true for a recursion to be a *valid* recursion?
        - Be able to implement (in code) a recursive strategy to solve a problem (as in Lab 7).

    b. Be familiar with the different advantages and disadvantages of the different ADT's and implementations we have discussed in class: ArrayList, LinkedList, BST, Hashtable, Set.

    c. Know the classes and interfaces involved in creating a user-defined Collection (like a List) in a way that can make use of Collections sorting and searching methods. Review when the List and Random-access interfaces are (or should be) implemented and the reasons for using the class AbstractList. Be familiar with what you need to do to a list that you create so that it can work with the sort and search functions available in the Collections class.

    d. Know how to use an Iterator object to iterate through the elements of a list. Know the difference between the Iterable and Iterator interfaces. Know the role of Iterator in the use of the for each construct. Be familiar with the new (as of Java 8) forEach method – both how to use it and where it is defined

    e. Be familiar with the top level of the exception's hierarchy provided in Java. Understand the difference between errors, unchecked exceptions, checked exceptions, and runtime exceptions. Know the most common examples of each type.

    f. Understand the finally keyword and be able to think through the behavior of a code sample like the one given in the finally Exercise at the end of Lesson 12.

    g. Know which background data structures are typically used to implement Array Lists, Linked Lists, Binary Search Trees, Hashtables, and Sets.

    h. Understand how the hashCode function is used in a class to support the use of objects as keys in a hashtable. Understand how a hashtable transforms input keys to hashcodes to hashvalues (and know the difference between these). Know why equal objects must have equal hashCodes and why it is *desirable* for unequal objects to have different hashCodes. Be familiar with best practices concerning the creation of a hashCode. Make sure you can write

code to override hashCode in any class that you create.

i. Given a sequence of ordered values (like integers or Strings), be able to follow the insertion rules to insert them into an initially empty BST.

j. Be able to write the code for a linked list implementation. Be sure that you are familiar with the technique for iterating from a top node (like a header) to some target node in the list. You may be asked to use Nodes to implement other data structures.

k. Be able to load a BST by hand using an insertion sequence

l. Understand how each of the following types of data structures is designed and implemented (in a general way): array lists, linked lists, stacks, queues hashtables, bsts, hashsets. Note that stacks and queues can be implemented in more than one way.

m. Be able to explain why, for ordering objects, sometimes the Comparable interface is not enough.

n. Be able to explain what it means for a Comparator to be consistent with equals, and why comparators *should be* consistent with equals. Be able to create your own Comparator so that it is consistent with equals and be able to use it in a call to Collections.sort.

o. Be able to use the new forEach default method of Iterable.

2. **Programming Questions.** There will be three programming questions:

    a. *Implement a data structure*. You will be given the type of background data structure to use; you will use that background data structure to implement some or all the main operations of the main data structure. (SinglyLinkedList, Queue, or Stack using Node)

    b. *Solve a problem using recursion*. This problem will be similar to problems you did in Lab 7.

    c. *Solve a problem using Polymorphism.* This problem will be like a midterm exam.

3. **SCI.** You will be asked to write a short essay to explain how one or more SCI principles are displayed in the realm of computer science. Richer content will be awarded more credit.