CSCE 314 [Sections 595, 596, 597] Programming Languages, Spring 2024

Hyunyoung Lee

Final Exam Preparation Guide

The exam will be held in-person on paper on Friday, May 3, 2024.

Please check the time and the room number for your section carefully.

Section	Room	Time	Date
595	**ZACH 315**	**5:45–6:35 PM**	**Friday, May 3**
596	**ZACH 340**	**6:50–7:40 PM**	**Friday, May 3**
597	**ZACH 353**	**7:55–8:45 PM**	**Friday, May 3**

Please bring

- **correct Scantron** (M-O 101807 gray form, size 8.5 × 11 in.), pick up the free Scantron, see https://provost.tamu.edu/office-of-the-provost/scantron-giveaway.html,
- **No.2 pencils and good erasers**, and
- **your ID**

The topics we covered in this course can be grouped into three categories:

- Haskell
- Fundamental topics on programming language design and implementation including grammars and type system
- Java

The final exam is comprehensive. Approximately 35% of the problems in the final exam will be related to Haskell, about 15% related to programming language design and implementation including grammars, and the rest 50% related to Java.

Thus, studying your midterm exam carefully is very important. Not only correctly answer each question, but be able to *explain* every question and every answer choice (why each choice is correct or not if it is a multiple choice question). Furthermore, study all of the exercises and quizzes in a similar way. Definitely, along the way, (assuming that you have already done the relevant reading of the textbooks) study using my lecture slides reviewing the material related to each question.

In particular, you are expected to know/understand the following topics and be able to answer the questions on them. Below, the first parts of the list of topics are overlapping with the midterm exam review sheets (they are included here for completeness).

1. [haskell-01-basics.pdf] Haskell basics: What does each technical term mentioned in the following sentence mean? *"Haskell is a lazy pure functional language."* Need to be familiar with the standard Prelude functions (at least the ones mentioned in the ten sets of lecture notes). Know how to apply the functions and what are the evaluation results.

   EQ1. Haskell's _____ is what helps guarantee referential transparency.

2. [haskell-02-types.pdf, haskell-03-functions.pdf] Haskell types, function types, type classes, and currying: Know how to define functions using conditional expressions, guarded equations, lambda expressions, `case` expressions, `let...in` expressions, and `where` blocks.

   EQ2. Consider a function `safetail :: [a] -> [a]` that behaves in the same way as the library function `tail`, except that `safetail` maps the empty list to the empty list, whereas `tail` gives an error in this case. Define `safetail` using

   (a) a conditional expression,
   (b) guarded equations, and
   (c) argument pattern matching.

   For (a) and (b), you need to make use of the library functions `tail` and `null`. (Hint: The library function `null :: [a] -> Bool` tests if a list is empty.)

3. [haskell-02-types.pdf, haskell-03-functions.pdf] Be able to tell the types of given Haskell expressions or functions. On currying, given a function definition with more than one arguments, you need to be able to give the correct type and explain how the curried version is understood in terms of lambda expressions.

   EQ3. What is the type of the following Haskell expression? `("Howdy,":"all":[])`

   EQ4. What is the type of `f`? `f x = x ‘div‘ 2`

   EQ5. What is the type of the following lambda expression? `\x y -> x <= y`

4. [haskell-03-functions.pdf] List comprehensions and recursive functions: Be able to reason the step-by-step of any recursive function given, for example, `quicksort`. What value a Haskell expression or a function application evaluates to.

   EQ6. Let `xs = [3,2,1]`. What is the result of the following expression?

   `zip xs (tail xs)`.

5. [haskell-04-higher.pdf] Higher-order functions: Be able to tell the types and meanings of each higher-order function covered in the lecture notes and how/where to use it.

   EQ7. Given the following definition for `g`, `g = (filter even) . (map (+1))`,

   (a) what is the type of `g`?
   (b) what is the result of `g [1..5]`?

6. [haskell-05-declaring-types.pdf] About programmer-defined data types: Defining functions involving programmer-defined types such as treeFold, the types of such functions, the evaluation steps (you should be able to show and explain those!) and results of such functions, deriving from type classes, etc.

EQ8. Given the following definition, `data Person = Person String Int`, what is the type of the constructor `Person`?

7. [haskell-06-modules.pdf] Defining and using the modules and the reasons of them.

   EQ9: Example ADT (Abstract Data Type) `Stack` with two different ways of implementing it and example uses of those stacks (as you practiced in the exercise).

8. [haskell-07-io.pdf] IO monad. Understand the workings of the basic IO actions such as `getChar`, `putChar`, and `return`, and the type of each of them. Also, understand the `do` (sequencing) notation to combine a sequence of actions together and the workings of the sequenced actions, for example, `act1` on slide #9, the three derived primitives on slide #10 (`getLine`, `putStr`, and `putStrLn`), and `main` on slide #12.

   EQ10. What does each of the three basic IO actions do? Also, what is each of their type?

   EQ11. What is the working of `act1` (with the definition on slide #9) for example?

9. [haskell-08-ch12-monads.pdf] How to make a data structure an instance of `Functor`, `Applicative` and `Monad`. Understand the application of the class member functions such as `fmap`, `pure`, `<*>`, and `>>=` to the corresponding objects and functions. Being able to explain step-by-step the workings of such applications.

   EQ12: (See the exercise questions)

10. [lec01-tour-of-lang-impl.pdf] Implementing a programming language – how to undo the abstraction: what each step tasks are, what each step is called, what are input and output of each step, in what order the steps are performed, and what error can be reported in which step.

    EQ13. If interpreted, *right after* which step is the interpreter invoked?

    EQ14. Order the tasks for implementing a programming language according to the order of how the abstraction is undone. (See the exercise questions)

11. [lec02-syntactic-analysis.pdf, ambiguity.pdf] Grammars: The Chomsky Hierarchy, context-free grammars, regular grammars, ambiguity of grammars and ways of disambiguating an ambiguous grammar.

    EQ15: (See the exercise questions)

12. [java-classes.pdf] Java basics such as OO programming basics, and Java classes, interfaces, inheritance, substitutability by subtyping, subtype hierarchy, static and dynamic binding, dynamic dispatch.

13. [JVM.pdf (only up to slide # 18)] The Java Virtual Machine (JVM), in particular, be able to answer briefly the following questions.
    (a) What is JVM? What is the purpose of JVM?
    (b) What are bytecodes?
    (c) What does the JVM specification define?
    (d) What is the internal architecture of JVM?

(e) What is stored in each of runtime data areas in JVM? How are they managed/maintained?
(f) What is JVM's verification process and what does it verify?

14. [JVM.pdf, java-activationRecExample.ppt] JVM stacks and activation records. The step-by-step involving activation records and objects on heaps, etc. as explained in the animated PPT slides. Which line of code triggers what steps?

15. [java-nestedTypes.pdf] Java nested types, static vs. non-static nested classes, local and anonymous inner classes, and their properties, how to use them (i.e., how to create those objects and invoke their methods, etc.).

16. [java-generics-partA.pdf] Java generics motivation, generic classes, generic methods, type parameter constraints, and motivation (such as without them what goes wrong or what kind of error message?) and syntax of F-bounded quantification.

17. [java-generics-partB.pdf] Type theory relating to the complex types including function types – co/contra/invariance. Type variance in Java, what is safe/unsafe in which type – argument type and return type when overriding methods, arrays, Collections, etc. Java wildcards – motivation of using wildcards, bounded wildcards – upper bounds, lower bounds, how to use them; what is considered as a subtype of what involving (bounded) wildcards?

18. [java-concurrency.pdf (only up to slide # 34)] Java concurrency – processes and threads, basic program constructs for writing concurrent programs, thread states and events that triggers the state transitions; synchronization issues – race hazard, locks (and deadlock), and conditional objects to avoid deadlock; use of `synchronized` keyword; Java memory model.

19. [java-reflection.pdf] Java reflection, ways to obtain a `Class` object, use of reflection functionalities via a Class object.

20. Main coverage of textbooks:

    (a) Haskell textbook Chapters 1–8, 10, and 12
    (b) Java textbook Chapters 1–11, 14, and 16

Summary:

- Study the lecture slides and watch the lecture videos
- Study the exercises and quizzes
- Study the example codes posted on Canvas
- Study the midterm exam
- Get enough sleep!

Good luck!!