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

Hyunyoung Lee

Midterm Exam Preparation Guide

The Exam will be held in-person on paper on Monday, March 4, 2024.

Please check the time and the room number on the syllabus for your section.

Format and Logistics. There will be 17 multiple choice questions and two free response questions (with subquestions each). The exam will be closed-book and closed-notes; and **absolutely no electronics, including calculators**, are allowed. The exam paper will be provided. Only things you need to bring with you to the exam are pencils and/or pens and your ID (TAMU ID, driver's ID, or passport).

Coverage. The midterm exam will cover Chapters 1–8, 10, 12 in the textbook and all my slides including lang01-tour-of-lang-impl.pdf and lang02-syntactic-analysis.pdf, and my notes on ambiguity (ambiguity.pdf). Topics from the textbook Chapters 13, 15, 16, and 17 are not directly asked as exam questions.

The main topics that you need to study to prepare for the exam are listed below with the corresponding set of lecture slides given in brackets [ ]. Basically, the midterm exam will cover everything we've been discussing so far since the first week including (but not limited to) reading the textbook, watching my videos with my lecture slides, solving exercises and quizzes, and homework problems. A few example questions are given below (preceded by EQ).

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] Know the evolution of programming languages, what it means by syntax and semantics of a programming language, and 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)

Happy studying and good luck!