Csc344 Final Exam Structure and Notes

Name →

Instructions

1. The final exam will be given on Wednesday, May 11, from 10:30am to 12:30pm.
2. You may turn in your exam before the 120 minute mark, but should you still be working on the exam after 120 minutes, you will need to turn it in promptly, silently, when you are asked to do so.
3. Write your answers to each of the questions in the "Answer Areas" provided.
4. No cognitive artifacts are allowed (notes, papers, phones, machines, or anything of the kind), other than the exam paper and a writing stick.
5. No communication of any sort is allowed with another human, including your professor.

Question 1 - Racket Programming with Images

Programming with images in Racket. Study lessons 1 and 2 of the Racket/Lisp lessons, and Racket/Lisp programming assignments 1 and 2.

Question 2 - BNF

BNF Deginitions and concepts. BNF Derivations. Writing BNF grammars. Parse trees. Study lesson 3 of the Racket/Lisp/BNF lessons, and problem set 1.

Question 3 - The Concept of Lisp

Questions about the history and the nature of Lisp. About Lisp. Features of Lisp. Dotted pairs and S-expressions. The functions of historical lisp. Study Lesson 6 of the Racket/Lisp lessons.

Question 4 - Historical Lisp in Racket

Questions relating to the historical Lisp langauage, as seen through a Racket lens, pertaining to both concepts and code. Study Lesson 7 of the Racket/Lisp lessons, and parts (especially the first task) of programming assignment 3.

Question 5 - Basic List Processing in Racket

Referencers and Constructors. Simple definitions using a number of primitives. Study lesson 8 of the Racket/Lisp lessons, and parts (especially the second, third, and fourth tasks) of programming assignment 3.

Question 6 - Recursive List Processing in Racket

Recursive List Processing. Study lesson 9 of the Racket/Lisp lessons, and parts of programming assignment 4.

Question 7 - Higher Order Functions in Racket

Higher order functions (e.g., map, filter, foldr and foldl, sort). Study lesson 10 of the Racket/Lisp lessons, and parts of programming assignment 4.

Question 8 - The Concept of Prolog

Questions about the history and the nature of Prolog. Terms. Unification. Perspectives on the representation of knowledge in Prolog. Study lesson 2 of the Prolog lessons.

Question 9 - Map Coloring in Prolog

The map coloring problem and its solution in Prolog. Study lesson 3 of the Prolog lessons, and task 1 of Prolog programming assignment 1.

Question 10 - KB Programming in Prolog

Programming and querying KBs, with an emphasis on writing predicates that take the form of rules. Study lesson 4 of the Prolog lessons, and tasks 2 and 3 of Prolog programming assignment 1.

Question 11 - List Processing in Prolog

List processing in Prolog, with an emphasis on head/tail notation and recursive programming. Study lesson 5 of the Prolog lessons, and task 4 of Prolog programming assignment 1.

Question 12 - State Space Problem Solving and Prolog

State space problems solving concepts, and state space programming in Prolog. Study lesson 7 of the Prolog lessons, and Prolog programming assignment 2.

Question 13 - First Haskell: Data, Types, Expressions, Functions

Basic elements of Haskell programming. Study parts of lesson 1 of the Haskell lessons, and tasks 1, 2, and 3 of the Haskell programming assignment.

Question 14 - Haskell as a Sign of Functional Programming

Questions about functional programming, and questions about the history and nature of Haskel, and questions Study parts of lesson 1 of the Racket/Lisp lessons.

Question 15 - Recursive Functions in Haskell

Recursive programming in Haskell. Study lesson 2 of the Racket/Lisp lessons, and task 4 of the Haskell programming assignment.

Question 16 - List Comprehensions in Haskell

List comprehensions in Haskell. Study lesson 3 of the Haskell lessons, and task 5 of the Haskell programming assignment.

Question 17 - Higher Order Functions in Haskell

Higher order functions (e.g., map, filter, foldr and foldl, zipWith) in Haskell, and other Haskell functions that play nicely with them (e.g., zip). Study lesson 4 of the Haskell lessons, and tasks 6, 7 and 8 of the Haskell programming assignment.

Question 18 - Types, Static vs Dynamic Typing, and Type Inference

What is a data type? What is the difference between static typing and dynamic typing in a programming langauge? Can you talk about static typing and dynamic typing with respect to Lisp (Racket) and Haskell? With respect to Java? Can you channel Haskell in doing some relatively simple type inferencing? In finding your way to "yes", Google can be your friend. If you can find your way to being able to answer "yes" to each of these questions, you should be ready for this question.

Question 19 - Aspects of Runtime Memory Management

What is the run time stack? What is the heap? What does it mean to explicitly manage run time memory? What is garbage collection? Can you provide an exampe of a language, not featured in the course, in which explicit memory management is required? Can you provide an example of a language, featured in this course, which performs garbage collection (or, better, three such languages)? Study the resources associated with problem set 2. Also, consult with Google, if need be. If you can find your way to being able to answer "yes" to each of these questions, you should be ready for this question!

Question 20 - The Concept of Rust

Can you say a few things about the Rust language, comparing it and contrasting it to Lisp, Prolog, and Haskell?. Can you talk about questions surrounding the adoption of Rust within the realm of professional sofware development? Study the resources associated with problem set 2.