Programming Languages Final Exam

Charlie Coleman

1 Functional Programming

1.1 What is a side effect, and why do functional languages not have them?

Side effects are observable effects outside of the return of a function, like outputting text to the console or modifying a file. This makes formal verification of code easier, and allows the output of code to be more easily predicted.

1.2 Give an example of how control flow is different in functional languages (particularly in Haskell). What type of statements in standard programming languages are not allowed in functional languages?

There is no for loop or while loop in Haskell. Everything is executed using a function call.

1.3 What is a higher order function? What is a first class object?

Higher order function — A function that takes another function as an argument or returns a function.
First class object — An object with no restrictions on its use. It is an object that can be passed and returned to/by functions.

1.4 How is I/O accommodated in functional programming languages, since it is pretty much purely based on side effects?

I/O is only allowed in specific places within Haskell, like a main function or a larger I/O block

1.5 What is a functor in Haskell?

Functor is a type class in Haskell. It is designed to hold things that can be mapped over, like lists.

1.6 How are types different in Haskell? Describe its type classes, and how they are different from object oriented classes.

New data types can be defined as composites of existing types. Type classes define what operations can be performed on different data types.

1.7 Be prepared to code Haskell functions, at the level of one of our homework assignments

1.8 What types of tools from functional programming are starting to show up in languages like Python and C++, and why are they being increasingly used there?

Map, filter, reduce — clean logic, parallelizable (especially map & filter)
First class objects — everything is an object in Python
Higher order function — useful for callback functions and other use cases
Lambda expressions — quick, single use functions prove useful in Python

1.9 How are the tools in Python or C++ different from a more traditional functional language, like Python or Lisp?

More complicated syntax, some require libraries to achieve the functionality,

1.10 Why do the more extreme proponents of functional programming object to structured paradigms? List a few examples from your reading on the subject (for that last essay).

2 Prolog

2.1 List a few applications of Prolog, or things that it can do well

AI, Natural Language Processing, good for enumeration of all possible solutions to a problem

2.2 What is unification, and how does Prolog attempt to do it?

Unification — applying the resolution principle and pattern matching things into appropriate spots
Rules of unification:

1. A constant only unifies with itself
2. Two structures unify if they have the same functor and arity, and the corresponding arguments unify recursively.
3. A variable unifies with anything. If the other thing has a value, then the variable is instantiated. If the other thing is an uninstantiated variable, then the two are associated so that later values will be shared.

2.3 What is a functor in Prolog?

A functor is the name outside the parenthesis of an operation or other data manip. If you say `rainy(seatle)` rainy is the functor.

2.4 How is a variable represented in Prolog? How are clauses formed?

Variables are all caps. Clauses are what define the functor output.

```
<base clause> ::= <structure>.
<non-base clause> ::= <structure :- <structures>.
```

2.5 Does the ordering of the clauses in a database matter in Prolog? Why or why not?

Yes, the clauses are checked in the order they are typed.

2.6 What is the cut (!) in Prolog?

Cut prevents prolog from backtracking past a goal. This essentially locks variables in their current state.

2.7 Again, be prepared to write or expand a short Prolog program, at the level of a single homework question or one of our examples from class.

3 COBOL

3.1 Name one place COBOL is still in use, and give several reasons.

A lot of places. Many businesses maintain legacy COBOL. This is because the codebases are so large and still function, so many companies don't think it worthwhile to update.

3.2 What are the principle strengths and weaknesses of COBOL?

Easy to read syntax. Can handle large amounts of data. Reliable.
No pointers, user defined types, user defined functions (initially)

3.3 How are COBOL programs organized?

1. Identification Division (required)
   — Supplies information about the program to the programmer and compiler
2. Environment Division
   — Used to describe the environment in which program should run
3. Data Division
   — Provides description to the data items the program will process
4. Procedure Division (required)
   — Contains code used to manipulate the data

4 Educational Languages

4.1 What are some of the major innovations that show up in educational programming languages?

Graphical representation of the code, syntax error prevention by only allowing valid commands

4.2 What are a few programming languages that have been developed for teaching purposes, and what are some of their strengths and weaknesses?

Alice — 3D environment, use is limited mostly to games and video creation.
Scratch — 2D environment, no first class functions, limited file I/O, can interact with Mindstorm