# CS 380 Outcome Statements

### AR/Machine Level Representation of Data

5. Describe the internal representation of non-numeric data, such as characters, strings, records, and arrays.

[Familiarity]

### PL/Object-Oriented Programming

4. Compare and contrast (1) the procedural/functional approach (defining a function for each operation with

the function body providing a case for each data variant) and (2) the object-oriented approach (defining a

class for each data variant with the class definition providing a method for each operation). Understand

both as defining a matrix of operations and variants. [Assessment]

5. Explain the relationship between object-oriented inheritance (code-sharing and overriding) and subtyping

(the idea of a subtype being usable in a context that expects the supertype). [Familiarity]

### PL/Functional Programming

1. Write basic algorithms that avoid assigning to mutable state or considering reference equality. [Usage]

2. Write useful functions that take and return other functions. [Usage]

4. Correctly reason about variables and lexical scope in a program using function closures. [Usage]

5. Use functional encapsulation mechanisms such as closures and modular interfaces. [Usage]

### PL/Event-Driven and Reactive Programming

2. Explain why an event-driven programming style is natural in domains where programs react to external

events. [Familiarity]

### PL/Basic Type Systems

1. For both a primitive and a compound type, informally describe the values that have that type. [Familiarity]

2. For a language with a static type system, describe the operations that are forbidden statically, such as

passing the wrong type of value to a function or method. [Familiarity]

3. Describe examples of program errors detected by a type system. [Familiarity]

4. For multiple programming languages, identify program properties checked statically and program

properties checked dynamically. [Usage]

7. Explain how typing rules define the set of operations that are legal for a type. [Familiarity]

8. Write down the type rules governing the use of a particular compound type. [Usage]

12. Explain multiple benefits and limitations of static typing in writing, maintaining, and debugging software.

[Familiarity]

### PL/Program Representation

1. Explain how programs that process other programs treat the other programs as their input data.

[Familiarity]

2. Describe an abstract syntax tree for a small language. [Usage]

3. Describe the benefits of having program representations other than strings of source code. [Familiarity]

### PL/Language Translation and Execution

2. Distinguish syntax and parsing from semantics and evaluation. [Familiarity]

4. Explain how programming language implementations typically organize memory into global data, text,

heap, and stack sections and how features such as recursion and memory management map to this memory

model. [Familiarity]

6. Discuss the benefits and limitations of garbage collection, including the notion of reachability. [Familiarity]

### PL/Syntax Analysis

1. Use formal grammars to specify the syntax of languages. [Usage]

3. Identify key issues in syntax definitions: ambiguity, associativity, precedence. [Familiarity]

### PL/Compiler Semantic Analysis

2. Describe semantic analyses using an attribute grammar. [Usage]

### PL/Runtime Systems

2. Discuss benefits and limitations of automatic memory management. [Familiarity]

3. Explain the use of metadata in run-time representations of objects and activation records, such as class

pointers, array lengths, return addresses, and frame pointers. [Familiarity]

### PL/Advanced Programming Constructs

• Lazy evaluation and infinite streams

• Control Abstractions: Exception Handling, Continuations, Monads

• Object-oriented abstractions: Multiple inheritance, Mixins, Traits, Multimethods

- 164 -

• Metaprogramming: Macros, Generative programming, Model-based development

• String manipulation via pattern-matching (regular expressions)

• Dynamic code evaluation (“eval”)

### PL/Formal Semantics

2. Write a lambda-calculus program and show its evaluation to a normal form. [Usage]

### PL/Language Pragmatics

1. Discuss the role of concepts such as orthogonality and well-chosen defaults in language design.

[Familiarity]

4. Show uses of delayed evaluation, such as user-defined control abstractions. [Familiarity]

### SDF/Fundamental Programming Concepts

8. Describe the concept of recursion and give examples of its use. [Familiarity]

9. Identify the base case and the general case of a recursively-defined problem. [Assessment]