CSE 307 – Principles of Programming Languages, Exam 02, 11/21/2019

Name:_____

SBUID: Key

Score (out of 120):_____

Directions:

(i) For short answer questions keep your responses concise; 1 or 2 sentences is often sufficient.
(ii) For True/False questions, please place a "T" or an "F" in the space provided to the left of each claim.
(iii) Otherwise, follow the directions given by the question.

01. Briefly explain the difference between *syntax* and *semantics* for programming languages.

Syntax is the form of a valid program

Semantics specify the meaning of a program (usually in terms of behavior)

02. Which of the following statements about programming language syntax and semantics are true?

T A. All syntactic rule violations are detected by the compiler.

F B. All semantic rule violations are detected by the compiler.

T C. All static semantic rule violations are detected by the compiler.

T D. Syntactic rules can be entirely specified as a Context-Free Grammar.

F E. Semantic rules can be entirely specified as a Context-Free Grammar.

03. How are dynamic semantic rules of a programming language enforced?

Code is added to the program by the compiler to check that all Semantic rules are followed

04. Give an example of a semantic feature of a programming language that is impossible to capture in a Contex-Free Grammar.

Type Consistency

# of arguments in Subroutine call matches # of parameters in Subroutine definition

05. Which of the following semantic checks can be performed as part of *static* analysis (Mark your selections with a check)?

- ✓ A. Type
- ✓ B. Alias
- _____ C. Morphological
- ✓ D. Escape
- ✓ E. Subtype

06. Please define the *scope* of a binding of a name and the thing it names.

The textual region of a program in which the binding is active.

07. Which of the following statements about names, scopes, and bindings are true?

- F A. All binding times are static.
- F B. All binding times are dynamic.
- F C. All bindings are between variables and values of basic types.
- T D. If a binding outlives its object, then this creates a dangling reference.
- F E. If an object outlives all its bindings, then this creates a dangling reference.

08. For each item below give the most likely *storage allocation mechanism* used to manage its space in memory. Enter "C" for static, "K" for stack, and "H" for heap.

H A. A linked-list node, where node data and list size can both be arbitrarily large.

C B. Code

K C. Local variables or C

C D. Global variables

K E. Return values or C

09. Explain the distinction between determining the scope of a binding *statically* vs. *dynamically*.

Static Scope is determined lexically by the relative position of variable uses to variable definitions

Dynamic Scope is determinid by relation between variable use and variable definition in history of program execution.

10. Which of the following statements about scoping are true?

T A. Bindings in one scope can be hidden by bindings in another scope.

F B. Every programming language has at least two scopes.

T C. Languages with static scoping typically follow a *most closely nested* scoping rule to determine the active binding for a name.

F D. Languages with dynamic scoping typically follow a *most closely nested* scoping rule to determine the active binding for a name

T E. Dynamic scoping is easier to implement than static scoping.

11. Name 5 ordering mechanism that can be used to determine the *control flow* of program execution.

Sequencing

Selection / Alternation (If)

Iteration (for / while)

Procedural Abstraction (Subroutines)

Recursion

Concurrency

Exception Handling

Nondeterminacy

12. Which of the following statements about control flow are true?

F A. Functional and Logic languages primarily rely upon sequencing to determine the control flow.

F B. Control flow is concerned only with statement execution; evaluation of expressions is irrelevant.

T C. Use of GOTO statements is a sign of unstructured control flow.

T D. Imperative languages primarily rely upon sequencing to determine control flow.

F E. When we assign a value to a variable we use the variable's r-value.

13. Please define the term *side effect*.

A permanent change made to the state of a program during execution.

- For subroutines, any changes or effects other than the return value.

14. Which of the following statements about data types and type systems are true?

T A. Strong typing prevents operations from being applied to inappropriate data types.

F B. Weak typing prevents operations from being applied to inappropriate data types.

F C. All strong typing is static typing.

F D. Strongly typed languages cannot also have polymorphism.

T E. Weakly typed languages like Perl will try to convert data into appropriate types.

15. Explain the distinction between the *Denotational* and *Structural* approaches to understanding data types.

Denotational - A type is identified with a set of values, e.g., to be of type integer is to be a member of the set of integers

Structural - (I meant constructive)
- Type determined by its composition of primitive or basic components

16. Given the following type definitions and variable declarations:

```
type student = record
name : string
address : string
age : integer

type school = record
name : string
address : string
age : integer

x : student
y : school
```

Are these two types equivalent?	Can a value of type `school` be assigned to a variable of type `student`?

A. How would these questions be answered for a language that uses *structural equivalence* for types?

Yes | Yes

B. How would these questions be answered for a language that uses *name equivalence* for types?

No | No

17. Given the following code:

```
z = object()
z.foo = 34
def f(a):
a = z
x = object()
x.foo = 11
f(x)
print x.foo
```

What is displayed by the print statement, `print x.foo`, for:

A. Call-by-value: 11

B. Call-by-reference: 34

C. Call-by-sharing: 11

18. Which of the following statements about calling *subroutines* are true?

F A. The caller is entirely responsible for executing the calling sequence.

F B. The callee is entirely responsible for executing the calling sequence.

T C. Ideally, the only registers that must be saved are ones currently used by the caller that the callee will overwrite.

F D. Arguments and local variables of a subroutine have a fixed address on the stack.

T E. The value of the stack pointer register must be changed when a subroutine call is executed.

19. Explain 2 different ways that a programming language can indicate the return value of a subroutine.

Specify with keyword "return"

pass return value to subroutine name

pass return value to designated variable / special return location

In Eiffel, its called "Result"

20. Which of the following statements about *object orientation* are true?

T A. Object-oriented data abstraction defines types in terms of the operations that the type supports.

F B. All fields of a parent class are accessible from its child classes.

T C. All public fields of a parent class are accessible from its child classes

F D. Java uses static method binding to provide subtype polymorphism.

T E. A class can have multiple constructors.

21. State the 3 key features of object-oriented programming.

Encapsulation

Inheritance

Dynamic Method Binding

22. Which of the following statements about *functional* programming languages are true?

T A. Functional languages are based upon the lambda calculus model of computation.

F B. Functional languages are based upon the Turing Machine model of computation.

T C. Functional languages are difficult to implement efficiently on von Neumann machines.

F D. Functional languages rely heavily on side effects for computation.

T E. Functional languages replace iteration with recursion.

23. Consider the following function definition in Scheme:

```
(define (f1 L1 L2) (if (null? L1) L2 (cons (car L1) (f1 (cdr L1) L2))))
```

Remember that:
`car` returns the head element of a list (equivalent to `hd` in SML)
`cdr` returns tail sublist of a list (equivalent to `tl` in SML)
`cons` adds an element to the front of a list (equivalent to `::` in SML)
The input to this function is two lists, L1 and L2. Explain what this function does.

Appends the list L1 onto the front of list L2 returning a single combined list.

24. Consider the following function definition in Scheme:

```
(define (f2 L) (if (null? L) '() (f1 (f2 (cdr L)) (list (car L)))))
```

The input to this function is a list, L. This function uses the function `f1` defined in question 23. Explain what this function does.

Takes a list L and returns the reverse of L as a list.