Midterm

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Started: Jun 27 at 3:05pm

Quiz Instructions

During this midterm test, you may access

1. Notes that you have taken during class (handwritten *or* electronic)
2. Study aides that *you* (or your study partners) have created (e.g., quizlet, evernote, studyblue, goconqr, tinycards, etc)
3. The textbook
4. The Daily PLs (including anything linked therefrom)
5. Class code (available at **https://github.com/hawkinsw/cs3003 (https://github.com/hawkinsw/cs3003)**)
6. Slides (original and annotated)

I trust that you will *not use the Internet to answer questions on this exam other than to access the materials above*. By submitting your exam responses through Canvas, you assert that you have not broken this trust. Violation of this accord will result in immediate and severe consequences.

With that out of the way, I know that you will do great on this test! Be sure to,

1. Read the questions closely,
2. Respond to *all* the aspects of the essay questions, and
3. Check your work.

If you have any questions during the test, please come see me! If you are in doubt about something, please ask! It's likely other students are having the same doubts!

There are 110 total points available but the exam is worth 100. In other words, 10 points of extra credit are available.

Good luck -- you won't need it!

Question 1 **2 pts**

________________________________ is the process of removing the details to simplify and focus attention on the essence. Some others like to say that ____________________ (*same word, no need to type it twice*) is all about remembering what's important and forgetting the rest (depending on the context, of course!).

abstraction

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 2

2 pts

A change between types done at the explicit request of the programmer is known as a(n) _______________.

explicit heap dynamic

Question 3

2 pts

In a(n) _______________________ scoped language, the scope of a variable can be determined by analysis of the program's source code.

static

Question 4

2 pts

____________________ define the effect of each statement's execution on the program's operation.

semantic

Question 5

2 pts

Edit View Insert Format Tools Table

12pt Paragraph B I U A

A(n) ____________ *(two words)* is a patte

a particular genre of programs and langua

programming paradigm

Question 6

2 pts

In a computer based upon the ____________________ (*two [or three] words*) model, instructions and data are stored together in the same memory.

Von Neumann

Question 7

2 pts

_______________ is the range of statements that can access a name binding.

scope

Question 8

2 pts

A variable's _______________ is the collection of its valid data values and the collection of valid operations on those values.

type

Question 9

Edit View Insert Format Tools Table

12pt Paragraph B I U A

A language with __________________________ (*two words*) is one without side effects.

referential transparency

Question 10

2 pts

In a language with __________________________ (*two words, first hyphenated*), evaluation of a Boolean expression is halted as soon as the result is completely determined.

short circuit evaluation

Question 11

2 pts

The __________________ *(two words)* of a statement contains all the name/variable bindings visible at that statement.

reference environment

Question 12

2 pts

A(n) ____________ is a subprogram that does not return any value(s).

Procedure

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 13

In ________________________ polymorphism, the subprogram's semantics are always the same.

parametric

Question 14

2 pts

A(n) ________ occurs when the language implicitly converts a variable of one type to another.

inference

Question 15

2 pts

A(n) _________________ is an association between an entity and an attribute.

binding

Question 16

2 pts

A dynamically typed language *cannot* be strongly typed.

○ True

[✓ False]

Edit View Insert Format Tools Table

12pt Paragraph

Question 17 2 pts

________ functions do not have side effects.

procedure

Question 18 2 pts

A(n) __________________ *(two words)* programming language is one where the type/variable binding is done before the code is run and does not change throughout program execution.

statically typed

Question 19 2 pts

______ is/are the rules for constructing a structurally valid computer programs in a programming language.

syntax

Question 20 2 pts

A variable's ____________________ is th
when it is associated with computer memo

Edit View Insert Format Tools Table

12pt Paragraph B I U A

lifetime

Question 21 4 pts

Order the steps of the processor/CPU's execution algorithm in a von Neumann Machine.

1	Fetch
2	Decode
3	Execute

Question 22 4 pts

Which of the following are considered by *Sebesta* to be valid binding times? (Select all that apply)

- ☐ Program design
- ☑ Program execution
- ☑ Language design
- ☑ Language implementation
- ☐ Program debugging

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 23

Which of the following are *true statements* about subprograms? (Select all that apply)

- [ ] All subprograms are expressions.
- [ ] A subprogram has multiple entry points.
- [x] Program execution returns to the caller upon completion.
- [x] Only one subprogram is active at a time.

Question 24

4 pts

Which of the following pieces of functionality are *required* for a PL is to be considered OO? (Select all that apply)

- [x] Dynamic binding
- [x] Abstract data types
- [x] Inheritance
- [ ] Open recursion
- [ ] Pointers

Question 25

4 pts

Which of the following are the valid lifetimes for a variable.

- [ ] Classic

- [x] Explicit heap dynamic
- [ ] Runtime
- [x] Stack dynamic
- [ ] Modern
- [x] Static
- [x] Implicit heap dynamic

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 26 5 pts

Consider this snippet of code (that *is **not** Python but shares a similar syntax!):*

```
def function_a():
    variable_v = "outer variable"
    def function_b():
        value_of_variable_v = variable_v
        print(f"variable_v: {value_of_variable_v}")

    def function_c():
      function_b()

    def function_d():
      variable_v = "not the outer variable."
      function_b()

    function_d()

def main():
    function_a()
```

In this *hypothetical* language,

1. Every function has its own scope (named the same as the function),
2. **scoping is static/lexical**, and
3. program execution begins with the `main` function.

When this program is invoked, execution begins at the `main` function which invokes `function_a`. In turn, `function_a` invokes `function_d`. `function_d` invokes `function_b`. Order the scopes that will be searched (from innermost to outermost) for `variable_v` when it is accessed in `function_b`.

1	function_b

2 functio

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 27

15 pts

You are designing a programming language that combines a runtime stack and the use of stack-dynamic storage for local variables to make it possible for programmers to write recursive functions. Describe how these two features interact so that a person writing an application with your language can use recursion.

In your answer, include responses to the following topics and/or questions:

1. What is the runtime stack?
2. Why recursion cannot be used when local variables are have a static lifetime?
3. At what point(s) in a program is the runtime stack modified?
4. What is the performance impact of using stack-dynamic storage versus static storage for local variables?
5. What is the space efficiency of using stack-dynamic storage versus static storage for local variables?

p

Edit View Insert Format Tools Table

12pt Paragraph B I U A

Question 28

15 pts

In class we discussed Algebraic Data Types (ADTs). ADTs come in two different flavors: sum types and product types. Please describe each of the two types. For each of the types you must

1. Choose a particular language and describe if/how that type is actually implemented. For the language that you choose, consider each of the following questions in your response:
   1. Does the language implement the sum/product type perfectly or must there be a *combination* of types that work together to accomplish the implementation (e.g., you add a *tag* to a data structure that the language supports natively)?
   2. In your opinion, does that language make it easy or hard to work with its version of the sum/product type? In other words, are the operations on sum/product types that we discussed in class easily expressed in the language?
2. Explain the reason that the Sum/Product type were given their names.
3. Describe a use-case that is particularly well suited for use by the Sum/Product type. In other words, describe a particular case where the availability of Sum/Product types makes code more expressive than it would be in the absence of such types.

Edit View Insert Format Tools Table

12pt Paragraph B I U A

p 0 words </>

Question 29

15 pts

Short-circuit evaluation plays a very important role in the semantics of the evaluation of Boolean expressions in a programming language. The user of a programming language needs to know if/how their language uses short-circuit evaluation. But, why?

Please describe the way that the presence/absence of short-circuit evaluation in a programming language can effect the behavior/correctness of a program written in a given language. To receive full points for the answer,

1. define short-circuit evaluation;
2. describe why designers might include short-circuit evaluation in their language;
3. explain how short-circuit evaluation can/does interact with other parts of the language's semantics to affect program behavior/correctness. Don't forget to address the characteristics of a programming language that would make it so that short-circuit evaluation *cannot* change the outcome of a program's behavior/correctness.

If it helps, feel free to refer of the differences in the potential behavior of the following two code (***pseudocode***) snippets in your answer:

Snippet A

```
# A function that does something and then returns
# true or false.
def sfn() -> bool:
    ...

# A function that does something else and then
# returns true or false.
def rfn() -> bool:
    ...

def main() -> None:
    if rfn() and sfn():
        ...
    if rfn() or sfn():
        ...
```

Snippet B:

Edit View Insert Format Tools Table

12pt Paragraph

```
# A function that does something and then re
# true or false.
def sfn() -> bool:
    ...

# A function that does something else and then
# returns true or false.
def rfn() -> bool:
    ...

def main() -> None:
    if sfn() and rfn():
        ...
    if sfn() or rfn():
        pass
```

Edit View Insert Format Tools Table

12pt Paragraph

1. A short-circuit evaluation of an expression is one in which the result is determined
without evaluating all of the operands and/or operators.

2. A reason why programmers might include short-circuit evaluation in their program is to optimize execution time. If the result of a Boolean expression can be determined in the first Boolean value, we can save time by not even considering the following value questions. For example, consider the expression: True or (one hundred more expressions that will take you 1 minute to execute...).

3. If there is a function with side-effects that is part of a Boolean expression, performing SCE on that expression will differ from not performing SCE on it.

p ▸ span 1 131 words </>

Quiz saved at 4:18pm Submit Quiz