# Midterm Sample Quiz

**Due** No due date **Points** 15 **Questions** 15 **Time Limit** None

**Allowed Attempts** Unlimited

# Instructions

**Attempt**

**Time**

**Score**

**LATEST**

[**Attem**](https://uc.instructure.com/courses/1625778/quizzes/5099687/history?version=1)

[**1**](https://uc.instructure.com/courses/1625778/quizzes/5099687/history?version=1)

[**pt**](https://uc.instructure.com/courses/1625778/quizzes/5099687/history?version=1)

15

out of

15

minutes

3

Submitted Oct 12 at 3:46pm

**1**

**/ 1 pts**

**Question 1**

In a(n) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ program

ming

language, the

*assignment*

statement (or expression) is akin to the

pipeline between the memory and the processor in the Von Neumann

Model.



imperative

**Correct!**

**orrect Answers**

object-oriented

imperative

Take this sample midterm quiz to get a sense of the types of questions that you will see on Friday! If you have any questions about the answers or the format, please contact me immediately!

[**Take the Quiz Again**](https://uc.instructure.com/courses/1625778/quizzes/5099687/take?user_id=4140280)

Attempt History

**1**

**/ 1 pts**

**Question 3**

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

Implicit heap dynamic

**Correct!**

Runtime

Modern

Classic

Explicit heap dynamic

**Correct!**

**/ 1 pts**

**1**

**Question 2**

In a

*statically typed programming language*

, all the variables must have

a type at the time of compilation. Please select

*all*

the different ways

that a variable can be given a type at the time of compilation. (Again,

please check

*all*

the ways!)

Implicit - Inference

**Correct!**

Implicit - Convention

**Correct!**

Assumed

Explicit

**Correct!**

Abstraction

Scoped

|  |
| --- |
| **Question 5 1 / 1 pts** |
| Consider this snippet of code*:*  **def**  wide  ()    {  receiver  =  "receiver contents"  **def**  out  ()    {  value\_of\_receiver  =  receiver  print  (  f  "receiver: {value\_of\_receiver}"  )  }  **def**  in  ()    {  out  ()  }  **def**  around  ()    {  receiver  =  "not the wide value"  out  ()    }  around  ()  }    **def**  main  ()    {  wide  ()    }  In this *hypothetical* language,  1. Every function has its own scope (enclosed between the { and the |

Stack dynamic

**Correct!**

Static

**Correct!**

**1**

**/ 1 pts**

**Question 4**

A language with \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (two words)

is one without side effects.



referential transparency

**Correct!**

**orrect Answers**

referential transparency

**1**

**/ 1 pts**

**Question 6**

Which of the following are characteristics of a subprogram? (Check all

that apply)

Subprogram has a single entry.

**Correct!**

Program execution returns to the caller upon completion of the

subprogram's execution.

**Correct!**

Caller is suspended until completion of the subprogram.

**Correct!**

Other Incorrect Match Options:

•

in

}

)

and the name of that scope is the same as the function,

.

2

**scoping is dynamic**

, and

.

3

program execution begins with the

main

function.

When this program is invoked, execution begins at the

main

function

which invokes

wide

. In turn,

wide

invokes

around

.

around

invokes

out

.

Order the scopes that will be searched (from innermost to outermost)

for

receiver

when it is accessed in

out

.

**1**

out

**Correct!**

**2**

around

**Correct!**

**3**

wide

**Correct!**

**4**

main

**Correct!**

|  |  |
| --- | --- |
| **Question 9** | **1 / 1 pts** |
| The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (two words) of a subprogram contains the number, order and type of its formal parameters. |  |

|  |
| --- |
| It cannot change the value of global variables. |

**1**

**/ 1 pts**

**Question 8**

The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is the time during which a variable is

bound to storage.



lifetime

**Correct!**

**orrect Answers**

lifetime

**1**

**/ 1 pts**

**Question 7**

The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ parameters of a subprogram are

declared in the subprogram header.



formal

**Correct!**

**orrect Answers**

formal



parameter profile

parameter profile

**1**

**/ 1 pts**

**Question 10**

A(n) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is a subprogram that does not return a

value.



procedure

**Correct!**

**orrect Answers**

procedure

**1**

**/ 1 pts**

**Question 11**

Subprograms are \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ abstractions.



process

**Correct!**

**orrect Answers**

process

|  |
| --- |
| **Question 12 1 / 1 pts** |
| The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of a language are the rules for constructing *valid* programs. |



syntax

syntax

**1**

**/ 1 pts**

**Question 13**

A(n)

*\_\_\_\_\_\_\_\_\_\_\_\_*

(

two words) is a pattern of problem-solving thought

that underlies a particular genre of programs and languages.



programming paradigm

**Correct!**

**orrect Answers**

programming paradigm

**1**

**/ 1 pts**

**Question 14**

A variable is \_\_\_\_\_\_\_\_\_ scoped to a unit or block of a program if it is

declared there.



locally

**Correct!**

**orrect Answers**

local

locally

|  |
| --- |
| **Question 15 1 / 1 pts** |
| The data structure that holds all the information about an active function is known as the *\_\_\_\_\_\_\_\_\_\_\_\_\_\_* (two words). |



stack frame

activation record

stack frame