Exam 1

Due Oct 1 at 3:20pm **Points** 60 **Questions** 17
Available Oct 1 at 2:20pm - Oct 1 at 3:20pm about 1 hour **Time Limit** 60 Minutes

Instructions

Content and timing

This exam will evaluate your understanding of all concepts discussed in the course thus far. It has 16 questions that you need to answer in one hour. You have one attempt at this exam.

General instructions

This is an open notes exam. You are free to look at your notes, any of the resources we used in class, or any website. However, you are not allowed to talk to your classmates or any other person to get help with the exam. You should not help your classmates with the exam or share any information about the exam to anyone. Any form of cheating will not be tolerated and will result in a grade of 0 for the exam and disciplinary action.

Exam-taking tips

I want you to complete the exam in time. Here are some tips that can help you answer it successfully.

- Answer questions that you think are easier first. Anything that takes you more than a minute to answer is "hard". Skip hard questions then come back for them later. This exam allows you to review and change your previous answers.
- Reading other questions can help you recall answers. In many cases, reading similar questions may help you answer other questions. Some questions may even offer hints that help you answer other questions. This gives you more reasons to skip "hard questions."
- Write the problem number in a piece of paper for review. This will help you review questions you skipped and ensure you answer all questions.
- Answer all questions. The chances of getting points for a question is 0 if you don’t provide an answer. It’s better to guess than to leave a problem blank.

Good luck!

This quiz was locked Oct 1 at 3:20pm.

Attempt History

	Attempt	Time	Score

	Attempt	Time	Score
LATEST	Attempt 1	49 minutes	57 out of 60

Score for this quiz: **57** out of 60
Submitted Oct 1 at 3:10pm
This attempt took 49 minutes.

Question 1

0 / 0 pts

By selecting "I agree", I indicate my understanding of the University Policies on Academic Honesty. I will exercise complete academic honesty by not giving or receiving any unauthorized assistance on this examination or engage in any form of cheating. I will not share any information about the exam to anyone. I also acknowledge that any confirmed act of dishonesty will result in the disqualification of this exam and appropriate sanctions, including formal conduct charges and possible course failure.

- I disagree
- [✓ I agree]

Correct!

Question 2

3 / 3 pts

What is the correct way of creating a String constant called id in Swift?

Correct!

let id: String

orrect Answers

let id: String

let id : String

Question 3

3 / 3 pts

What should we use as the argument label name in the function prototype if we don't want the user to provide a parameter name when calling it? For example, a user can call the display function as shown below.

```
var schoolAcronym = "CSUF"

display(schoolAcronym)
```

ou Answered

_ schoolAcronym: String

orrect Answers

_

Question 4

3 / 3 pts

Create the function prototype of a function called getFirstName that accepts a single String parameter called name and returns a String.

*Note: The function prototype only provides the name, parameters, and return type of a function. You can also think of it as the function without the body (curly braces and anything inside it.)

ou Answered

func getFirstName(name: String) -> String;

orrect Answers

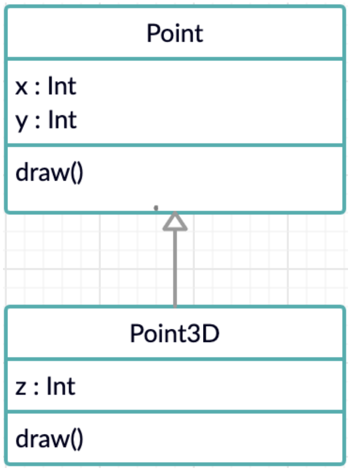
func getFirstName(name: String) -> String

func getFirstName(name : String) -> String

Question 5

3 / 3 pts

The diagram below describes the inheritance relationship between Point and Point3D. Is Point3D more likely to be a struct or a class?



Point3D is a struct

Correct!

[✓ Point3D is a class]

Point3D is neither a class nor a struct

Question 6

0 / 3 pts

Consider the code and the output below. Is CreditCard more likely to be a struct or a class?

```
func swipe(card: CreditCard, cost: Double) {
  card.charge(amount: cost)
}

var visa = CreditCard(limit: 5000)
print("Limit before swiping: $\(visa.limit)")
swipe(card: visa, cost: 100)
print("Limit after swiping: $\(visa.limit)")
```

Screen Output:

Limit before swiping: $5000.0
Limit after swiping: $4900.0

○ CreditCard is neither a class nor a struct

ou Answered [✓ CreditCard is a struct]

orrect Answer ○ CreditCard is a class

Question 7

10 / 10 pts

Complete the code below to create a structure called Color that stores the value of it's red, green, and blue components as properties. The red, green, and blue components are stored as Integer values. Make sure that the first property is called red, the second is called green, and the third is called blue.

```
struct Color {
  var red: Int
  var green: Int
  var blue: Int
}
```

Create a Color instance and set it's red to 255, green to 165, and blue to 0.

var orange : Color = Color(red: 255, gree

Answer 1:

Correct! struct

Answer 2:

Correct!

var red: Int

orrect Answer

var red : Int

Answer 3:

Correct!

var green: Int

orrect Answer

var green : Int

Answer 4:

Correct!

var blue: Int

orrect Answer

var blue : Int

Answer 5:

Correct!

Color(red: 255, green: 165, blue: 0)

orrect Answer

Color(red : 255, green : 165, blue : 0)

Question 8

3 / 3 pts

Complete the code by providing a conditional statement that checks a fruit's weight and color. It is only an apple if the fruit's color is red, and its weight is between 70 and 100 grams (inclusive).

```
func isApple(color: String, weight: Int) {
 if ( (color == "red") && {
   print("It's an apple!")
 } else {
   print("It's not an apple :(")
 }
}
```

Answer 1:

ou Answered ((color == "red") && (weight >= 70 && weight <= 100))

orrect Answer color == "red" && weight >= 70 && weight <= 100

orrect Answer color == "red" && weight > 69 && weight < 101

Question 9

2 / 2 pts

What keyword should you add to a structure's member function so it can modify its properties? The compiler will produce an error when it is missing.

Correct! mutating

orrect Answers mutating

Question 10

5 / 5 pts

Which among the following are valid case statements for the switch statement below?

```
var numGitCommits = 162
switch numGitCommits {
 // case statements here
}
```

Correct! ✓ default: print("Someone should be a senior developer...")

Correct! ✓ case 0: print("Time to make some changes!")

Correct! ✓ case 1...30: print("Good progress!")

Correct!

[✓] case 31..<100: print("Impressive")

[ ] case <150: print("Junior developer")

Consider the code below to answer the four questions that follow

```
struct MovingPoint {
  var prevX : Int = 0
  var prevY : Int = 0

  var x: Int {
    willSet {
      prevX = x
    }
  }
  var y: Int {
    willSet {
     prevY = y
    }
  }

  var location : String {
    return "Point moved from (\(prevX), \(prevY)) to (\(x), \(y))"
  }
}

var target = MovingPoint(x: 5, y:10)
target.x = 10
target.y = 20
print(target.location)
```

Screen output:

```
Point moved from (5, 10) to (10, 20)
```

Question 11

3 / 3 pts

Which among the following properties is a computed property?

*Note: Refer to the MovingPoint class in the previous instructions

Correct!
- [✓ location]
- y
- prevY
- x
- prevX

Question 12

3 / 3 pts

Which among the following properties have a property observer?

*Note: Refer to the MovingPoint class in the previous instructions

Correct!
- [✓ y]
- location
- prevY
- prevX

Correct!
- [✓ x]

Question 13

3 / 3 pts

How did the prevX and prevY variable values change to 5 and 10?

*Note: Refer to the MovingPoint class in the previous instructions

○ The computed property for x creates a reference between prevX and x that allows it to access its old value. The computed property for y creates a reference between prevY and y that allows it to access its old value.

○ The property observer for x, copied the new value of x to prevX. The property observer for y, copied the new value of y to prevY.

Correct!

[✓ The property observer for x, copied the previous value of x to prevX. The property observer for y, copied the previous value of y to prevY.]

○ Their values were assigned to 5 and 10 when the MovingPoint instance was created.

Question 14

3 / 3 pts

What will happen when we run the following unit test on the MovingPoint structure? Assume that we are using the XCTest framework for testing.

*Note: Refer to the MovingPoint class in the previous instructions

```
func testMovement() {
 var target = MovingPoint(x: 1, y:1)
 target.x = 3
 target.y = 6

 target.x = 8
 target.y = 15

 XCTAssertEqual(target.prevX, 3)
 XCTAssertEqual(target.prevY, 6)
}
```

○ This is not a valid unit test because the function name should start with unitTest.

Correct!

[✓ The unit test will pass.]

○ The unit test will fail.

○ The unit test will not run because you can only have one assertion inside a unit test function.

Question 15

4 / 4 pts

Match the following scenarios with **the most appropriate** data structure to use. The most appropriate data structure is one that naturally solves the problem, does not require additional statements to work, and often results in shorter code.

Correct!

You want to store the the amount of time it takes you to cook dinner for 5 days. You will use this information to compute for the average time it takes you to prepare dinner.

array

Correct!

You want to keep track of the number of times your favorite tennis player won a match. You will use this information to identify the player with the most wins.

dictionary

Other Incorrect Match Options:

- tuple
- class
- structure

Question 16

4 / 4 pts

Complete the program below so that it displays all Anime that were released after 1990.

```
struct Anime {
 var title: String
 var releaseDate: Int
}

var animeList = ["Dragon Ball" : 1984, "Naruto" : 2002, "Ranma" : 1987, "Fushigi Yūgi" : 1995, "Sailor Moon" : 1992]

for (title, releaseDate) i {

 if releaseDate > 1990 {
   print("\(title) (\(releaseDate))")
 }
}
```

Answer 1:

Correct!

(title, releaseDate) in animeList

orrect Answer

(title , releaseDate) in animeList

Question 17

5 / 5 pts

Based on your experience with Swift so far, identify at least one Swift feature that you like compared to other programming languages you have used.

Your Answer:

Property observers and computed properties are easily two of my favorite features within swift that I am not aware of other languages having. I find it really useful when I want a certain property to automatically update itself depending on a preset condition. Especially, if I want a property to be computed automatically, it's very nice and automated, rather than having to call a member function to do the same job. Definitely a nice functionality.

Quiz Score: **57** out of 60