####Q1. What is this code an example of?

```

let val = (Double)6

```

- [x] an error

- [ ] typecasting

- [ ] assignment

- [ ] initialization

####Q2. What is the error in this code?

```

let x = 5

guard x == 5 { return }

```

- [x] The guard is missing the else.

- [ ] Nothing is wrong.

- [ ] The guard is missing a then.

- [ ] The comparison is wrong.

####Q3. What is the raw/underlying type of this enum?

```

enum Direction {

case north, south, east, west

}

```

- [x] There is none.

- [ ] String

- [ ] Any

- [ ] Int

####Q4. Why is dispatchGroup used in certain situations?

- [x] It allows multiple synchronous or asynchronous operations to run on different queues.

- [ ] It allows track and control execution of multiple operations together.

- [ ] It allows operations to wait for each other as desired.

- [ ] all of these answers.

####Q5. What is this code an example of?

```

let val = 5

print("value is: \(val)")

```

- [x] string interpolation

- [ ] string compilation

- [ ] method chaining

- [ ] string concatenation

####Q6. What are the contents of vals after this code is executed?

```

var vals = [10, 2]

vals.sort { (s1, s2) -> Bool in

s1 > s2

}

```

- [x] [10, 2]

- [ ] [2, 10]

- [ ] nil

- [ ] This code contains an error

####Q7. What does this code print?

```

typealias Thing = [String, Any]

var stuff: Thing

print(type(of: stuff))

```

- [ ] Dictionary<String, Any> (To print this than code in question has to be `typealias Thing = [String: Any]`)

- [ ] Dictionary

- [x] ERROR (If code in question is really like that.)

- [ ] Thing

####Q8. What is the value of y?

```swift

let x = ["1", "2"].dropFirst()

let y = x[0]

```

- [x] This code contains an error

- [ ] 1

- [ ] 2

- [ ] nil

####Q9. What is the value of test in this code?

```swift

var test = 1 == 1

```

- [x] TRUE

- [ ] YES

- [ ] 1

- [ ] This code contains an error

####Q10. What is the value of y?

```swift

var x: Int?

let y = x ?? 5

```

- [x] 5

- [ ] 0

- [ ] nil

- [ ] This code contains an error

####Q11. What is the type of this function?

```swift

func add(a: Int, b: Int) -> Int { return a+b }

```

- [ ] Int

- [x] (Int, Int) -> Int

- [ ] Int<Optional>

- [ ] Functions don't have types.

####Q12. What is the correct was to call this function?

```swift

func myFunc(\_ a: Int, b: Int) -> Int {

return a + b

}

```

- [x] myFunc(5, b: 6)

- [ ] myFunc(5, 6)

- [ ] myFunc(a: 5, b: 6)

- [ ] myFunc(a, b)

####Q13. The Codable protocol is \\*\*\*\\*\\_\\*\\*\*\*?

- [x] a combination of Encodable and Decodable

- [ ] not a true protocol <<<<---Possibly correct as it's a typealias of Encodable and Decodable

- [ ] required of all classes

- [ ] automatically included in all classes

####Q14. What is the type of value1 in this code?

```swift

let value1 = "\("test".count)"

```

- [x] String

- [ ] Int

- [ ] null

- [ ] test.count

####Q15. When a function takes a closure as a parameter, when do you want to mark is as escaping?

- [x] when it's executed after the function returns

- [ ] when it's scope is undefined

- [ ] when is's lazy loaded

- [ ] all of these answers

####Q16. What's wrong with this code?

```

class Person {

var name: String

var address: String

}

```

- [x] Person has no initializers.

- [ ] Person has no base class.

- [ ] var name is not formatted corrrectly.

- [ ] address is a keyword.

####Q17. What is the value of names after this code is executed?

```

let names = ["Bear", "Joe", "Clark"]

names.map { (s) -> String in

return s.uppercased()

}

```

- [ ] ["BEAR", "JOE", "CLARK"]

- [ ] ["B", "J", "C"]

- [x] ["Bear", "Joe", "Clark"]

- [ ] This code contains an error.

####Q18. What describes this line of code?

```

let val = 5

```

- [x] a constant named val of type Int

- [ ] a variable named val of type item

- [ ] a constant named val of type Number

- [ ] a variable named val of type Int

####Q19. What is the error in this code?

```

extension String {

var firstLetter: Character = "c" {

didSet {

print("new value")

}

}

}

```

- [x] Extensions can't add properties. // although extensions technically can't contain stored properties

- [ ] Nothing is wrong with it.

- [ ] didSet takes a parameter.

- [ ] c is not a character.

####Q20. didSet and willSet are examples of `\\*\*\*\\*\\_\\*\\*\*\*`?

- [x] property observers

- [ ] key properties

- [ ] all of these answers

- [ ] newOld value calls

####Q21. What is wrong with this code?

```

self.callback = {

self.attempts += 1

self.downloadFailed()

}

```

- [x] Use of self inside the closure causes retain cycle.

- [ ] You cannot assign a value to closure in this manner.

- [ ] You need to define the type of closure explicitly.

- [ ] There is nothing wrong with this code.

####Q22. How many values does vals have after this code is executed?

```

var vals = Set<String> = ["4", "5", "6"]

vals.insert("5")

```

- [x] three

- [ ] four

- [ ] eight

- [ ] This code contains an error.

####Q23. How can you avoid a strong reference cycle in a closure?

- [x] Use a capture list to set class instances of weak or unowned.

- [ ] You can't, there will always be a danger of strong reference cycles inside a closure.

- [ ] Initialize the closure as read-only.

- [ ] Declare the closure variable as lazy.

####Q24. What is wrong with this code?

```

if let s = String.init("some string") {

print(s)

}

```

- [x] This String initializer does not return an optional.

- [ ] String does not have an initializer that can take a String.

- [ ] = is not a comparison.

- [ ] Nothing is wrong with this code.

####Q25. Which code snippet correctly creates a typealias closure?

- [x] typealias CustomClosure: () -> ()

- [ ] typealias CustomClosure { () -> () }

- [ ] typealias CustomClosure -> () -> ()

- [ ] typealias CustomClosure -> () {}

####Q26. How do you reference class members from within a class?

- [x] self

- [ ] instance

- [ ] class

- [ ] this

####Q27. All value types in Swift are \\*\*\*\\*\\_\\*\\*\*\* under the hood?

- [x] structs

- [ ] classes

- [ ] optionals

- [ ] generics

####Q28. What is the correct was to ass a value to this array?

```

var strings = [1, 2, 3]

```

- [x] all of these answers

- [ ] strings.append(4)

- [ ] strings.insert(5, at: 1)

- [ ] strings += [5]

####Q29. How many times will this loop be executed?

```

for i in 0...100 {

print(i)

}

```

- [ ] 0

- [x] 101

- [ ] 99

- [ ] 100

####Q30. What can AnyObject represent?

- [x] an instance of any class

- [ ] an instance of an optional type

- [ ] an instance of a function type

- [ ] all of these answers

####Q31. What can AnyObject represent?

- [ ] an instance of any class

- [ ] an instance of function type

- [x] all of these answers

- [ ] an instance of an optional type

####Q32. What does this code print?

```

typealias Thing = [String:Any]

var stuff : Thing

print(type(of:stuff))

```

- [ ] Dictionary

- [ ] ERROR

- [ ] Thing

- [x] Dictionary<String, Any>

####Q33. What is the value of test in this code?

```

var test = 1 == 1

```

- [x] TRUE

- [ ] 1

- [ ] This code contains an error.

- [ ] YES

####Q34. What is the value of y?

```

var x : Int?

let y = x ?? 5

```

- [ ] 0

- [ ] nil

- [ ] This code contains an error.

- [x] 5

####Q35. What is the value of y?

```

let x = ["1","2"].dropFirst()

let y = x[0]

```

- [ ] 1

- [ ] nil

- [x] This code contains an error.

- [ ] 2

####Q36. What is the value of t after this code is executed?

```

let names = ["Larry", "Sven", "Bear"]

let t = names.enumerated().first().offset

```

- [x] This code is invalid.

- [x] This code does not compile.

- [ ] `0`

- [ ] `1`

- [ ] `Larry`

####Q37. What is the value of test after this code executes?

```

let vt = (name: "ABC", val: 5)

let test = vt.0

```

- [x] ABC

- [ ] 0

- [ ] 5

- [ ] name

####Q38. What is the base class in this code?

```

class LSN : MMM {

}

```

- [x] MMM

- [ ] LSN

- [ ] There is no base class.

- [ ] This code is invalid.

####Q39. What does this code print to the console?

```

var userLocation: String = "Home" {

willSet(newValue) {

print("About to set userLocation to \(newValue)...")

}

didSet {

if userLocation != oldValue {

print("userLocation updated with new value!")

} else {

print("userLocation already set to that value...")

}

}

}

userLocation = "Work"

```

- [x] About to set userLocation to Work… userLocation updated with new value!

- [ ] About to set userLocation to Work… userLocation already set to that value…

- [ ] About to set userLocation to Home… userLocation updated to new value!

- [ ] ERROR

####Q40. What must a convenience initializer call?

- [ ] a base class convenience initializer

- [ ] either a designated or another convenience initializer

- [x] a designated initializer

- [ ] none of these answers

####Q41. Which object allows you access to specify that a block of code runs in a background thread?

- [ ] `DispatchQueue.visible`

- [x] `DispatchQueue.global`

- [ ] `errorExample` need to be labeled as `throws`.

- [ ] `DispatchQueue.background`

####Q42. What is the inferred type of x?

```

let x = ["a", "b", "c"]

```

- [ ] `String[]`

- [x] `Array<String>`

- [ ] `Set<String>`

- [ ] `Array<Character>`

####Q43. What is the value of `oThings` after this code is executed?

```

let nThings: [Any] = [1, "2", "three"]

let oThings = nThings.reduce("") { "\($0)\($1)" }

```

- [ ] `11212three`

- [ ] `115`

- [x] `12three`

- [ ] Nothing, this code is invalid.

####Q44. How would you call a function that throws errors and also returns a value?

- [ ] `!try`

- [x] `try?`

- [ ] `try!`

- [ ] `?try`

####Q45. What is wrong with this code?

```

protocol TUI {

func add(x1 : Int, x2 : Int) -> Int {

return x1 + x2

}

}

```

- [ ] Protocol functions cannot have return types.

- [x] Protocol functions cannot have implementations.

- [ ] Nothing is wrong with it.

- [ ] `add` is a reserved keyword.

####Q46. In this code, what are `wheels` and `doors` examples of?

```

class Car {

var wheels: Int = 4

let doors = 4

}

```

- [ ] class members

- [ ] This code is invalid.

- [ ] class fields

- [x] class properties

####Q47. How do you designate a failable initializer?

- [ ] `init?`

- [ ] `deinit`

- [x] `init`

- [ ] You can't.

####Q48. How do you designated a failable initializer?

- [ ] You cannot

- [ ] `deinit`

- [x] `init?`

- [ ] `init`

####Q49. What is printed when this code is executed?

```

let dbl = Double.init("5a")

print(dbl ?? ".asString()")

```

- [ ] `five`

- [ ] `5a`

- [x] `.asString()`

- [ ] `5`

####Q50. In the function below, what are \*\*this\*\* and \*\*toThat\*\* examples of?

```swift

func add(this x: Int, toThat y: Int)->{}

```

- [ ] none of these answers

- [ ] local terms

- [x] argument labels

- [ ] parameters names

####Q51. What is wrong with this code?

```swift

if let s = String.init("some string"){

print (s)

}

```

- [ ] Nothing is wrong with this code

- [ ] = is not a comparison

- [ ] String does not have an initializer that can take a String

- [x] This String initializer does not return an optional

####Q52. What is wrong with this code?

```swift

for (key, value) in [1: "one", 2: "two"]{

print(key, value)

}

```

- [ ] The interaction source is invalid

- [ ] The interaction variable is invalid

- [x] There is nothing wrong with this code

- [ ] The comma in the print is misplaced

####Q53. Which of these choices is associated with unit testing?

- [ ] XCTest

- [x] all of these answers

- [ ] @testable

- [ ] XCAssert

####Q54. In the code below, what is width an example of?

```swift

class Square{

var height: Int = 0

var width : Int {

return height

}

}

```

- [ ] This code contains error

- [ ] a closure

- [x] a computed property

- [ ] lazy loading

####Q55. What data type is this an example of?

```swift

let vals = ("val", 1)

```

- [ ] a dictionary

- [x] a tuple

- [ ] an optional

- [ ] This code contains error

####Q56. What is wrong with this code?

```swift

var x = 5

x = 10.0

```

- [x] You cannot assign a Double to a variable of type Int

- [ ] x is undefined

- [ ] x is a constant

- [ ] x has no type

####Q57. What is the type of x: let x = try?

```swift

String.init(from: decoder)

```

- [ ] String

- [x] String?

- [ ] String!

- [ ] try?

####Q58. What will this code print to the console?

```swift

var items = ["a":1, "b":2, "c":"test"] as [String: Any]

items["c"] = nil

print(items["c"] as Any)

```

- [ ] `Any`

- [ ] `test`

- [ ] `1,2,3`

- [x] `nil`

####Q59. What is wrong with this code?

```swift

let val = 5.0 + 10

```

- [x] There is nothing wrong with this code

- [ ] val is a constant and cannot be changed

- [ ] 5.0 and 10 are different types

- [ ] There is no semicolon

####Q60. How many parameters does the initializer for Test have?

```swift

struct Test{

var score: Int

var date: Date

}

```

- [ ] zero

- [ ] This code contains an error

- [x] two

- [ ] Structs do not have initializers

####Q61. What prints to the console when executing this code?

```swift

let x = try? String.init("test")

print(x)

```

- [ ] `nil`

- [ ] Nothing - this code contains an error

- [x] `Optional("test")`

- [ ] `test`

####Q62. How can you sort this array?

```swift

var vals = [1,2,3]

```

- [ ] `vals.sort { $0 < $1 }`

- [ ] `vals.sort { (s1, s2) in s1 < s2 }`

- [ ] `vals.sort(by: <)`

- [x] all of these answers

####Q63. What is printed when this code is executed?

```swift

let dbl = Double.init("5a")

print(dbl ?? ".asString()")

```

- [ ] `5a`

- [ ] `5`

- [ ] `five`

- [x] `asString()`

####Q64. DispatchQueue.main.async takes a block that will be

- [ ] not executed

- [x] executed in the main queue

- [ ] none of these answers

- [ ] executed on the background thread

####Q65. What is the value of test after this code executes?

```swift

let vt = (name: "ABC", val: 5)

let test = vt.0

```

- [x] `ABC`

- [ ] `name`

- [ ] `5`

- [ ] `0`

####Q66. When is deinit called?

- [ ] When a class instance needs memory

- [x] All of these answers

- [ ] When the executable code is finished

- [ ] When a class instance is being removed from memory

####Q67. How do you declare an optional String?

- [x] `String?`

- [ ] `Optional[String]`

- [ ] `[String]?`

- [ ] `?String`

####Q68. Why is dispatchGroup used in certain situation?

- [x] All of these answers

- [ ] It allows multiple synchronous or asynchronous operations to run on different values

- [ ] It allows operations to wait for each other as defined

- [ ] It allows track and control execution of multiple operations together

####Q69. How many times this code will be executed? —OR— How many times will this loop be performed?

```swift

for i in ["0", "1"]{

print(i)

}

```

- [ ] one

- [x] two

- [ ] three

- [ ] This code does not compile

####Q70. What does this code print?

```swift

let names = ["Bear", "Tony", "Svante"]

print(names[1]+"Bear")

```

- [ ] `1Bear`

- [ ] `BearBear`

- [x] `TonyBear`

- [ ] Nothing, this code is invalid

####Q71. What is true of this code?

```swift

let name: String?

```

- [ ] name can hold only a string value.

- [x] name can hold either a string or nil value.

- [ ] Optional values cannot be `let` constants.

- [ ] Only non-empty string variables can be stored in name.