4143 - PLC
Fall 2023
Exam One
October 19, 2023

Name: _____

1. (2 points) Which of the following is the Go convention for exported names?
    A. They start with an underscore.
    B. They start with a lowercase letter.
    C. They start with an uppercase letter.
    D. They end with an exclamation mark.
    E. None of the above

2. (2 points) If you see a Go function named ‘calculateTotal‘, what can you infer about its visibility?
    A. It’s exported and can be accessed from other packages.
    B. It’s exported but cannot be accessed from other packages.
    C. It’s a built-in Go function.
    D. It’s an anonymous function.
    E. None of the above

3. (2 points) Which of the following is a recommended naming convention for package names in Go?
    A. CamelCase
    B. snake_case
    C. all-lowercase
    D. ALL-UPPERCASE
    E. None of the above

4. (2 points) Given a struct attribute named ‘size‘, what would be the conventional getter method name in Go?
    A. ‘getSize()‘
    B. ‘Size()‘
    C. ‘GetSize()‘
    D. ‘retrievesize()‘
    E. None of the above

5. (2 points) For an acronym like ‘XML‘, how should it be represented in a Go identifier when combined with another word, e.g., "parser"?
    A. ‘xMLParser‘
    B. ‘XmlParser‘
    C. ‘XMLParser‘
    D. ‘xmlparser‘

E. None of the above

6. (2 points) In Go, which of the following is the conventional way to allocate memory and create a new instance of a struct?

   A. ‘new(MyStruct)‘
   B. ‘MyStruct.new()‘
   C. ‘&MyStruct‘
   D. ‘create MyStruct‘
   E. None of the above

7. (2 points) How does Go determine if a field or method of a struct is exported outside of its package?

   A. By using the ‘export‘ keyword.
   B. By placing the field or method in a separate ‘public‘ section.
   C. By annotating the field or method with ‘// public‘.
   D. By starting the field or method name with an uppercase letter.
   E. None of the above

8. (2 points) Which of the following is NOT a naming convention in Go?

   A. Starting a variable name with an underscore to indicate it’s unused.
   B. Using camelCase for variable names.
   C. Prefixing interface names with an ‘I‘ (e.g., ‘IReader‘).
   D. Using all uppercase letters for constant values.
   E. None of the above

9. (2 points) Which of the following best describes a slice in Go?

   A. An STL vector.
   B. A dynamic array that can grow or shrink.
   C. A linked list.
   D. All of the above
   E. None of the above

10. (2 points) What is the output of the following code?

```
s := []int{1, 2, 3, 4, 5}
t := s[1:4]
t[0] = 10
fmt.Println(s[1])
```

   A. 1
   B. 2
   C. 10
   D. An error is produced.
   E. None of the above

11. (2 points) How do you append an element to a slice in Go?
    A. ‘s.add(10)‘
    B. ‘s++‘
    C. ‘s + 10‘
    D. ‘s = append(s, 10)‘
    E. None of the above

12. (2 points) In Go, what’s the primary purpose of a struct?
    A. To define a new data type.
    B. To group together variables of different data types.
    C. To implement interfaces.
    D. To define methods.
    E. None of the above

13. (2 points) Which of the following is a correct way to initialize a struct named ‘Person‘ with fields ‘Name‘ and ‘Age‘?
    A. ‘p := Person(Name: "Alice", Age: 30)‘
    B. ‘p := PersonName="Alice", Age=30‘
    C. ‘p := PersonName: "Alice", Age: 30‘
    D. ‘p := new Person("Alice", 30)‘
    E. None of the above

14. (2 points) When you have a struct with an embedded struct, which statement is true?
    A. The embedded struct’s fields can be accessed as if they were fields of the outer struct.
    B. You must always use the embedded struct’s name to access its fields.
    C. The outer struct inherits all the methods of the embedded struct.
    D. All of the above
    E. None of the above

15. (2 points) Which of the following best describes a map in Go?
    A. A dictionary like Python
    B. An associative array like in Php
    C. An Stl Map like in C++
    D. A data structure that associates keys with values.
    E. All of the above

16. (2 points) How do you declare a new empty map with string keys and int values?
    A. ‘m := new(mapint)‘
    B. ‘m := mapint‘
    C. ‘var m mapint = ‘
    D. ‘var m = []stringint‘

E. None of the above

17. (2 points) What is the result of accessing a key that doesn't exist in a map?
    A. The program will crash with an error.
    B. It returns ‘nil‘.
    C. It returns the zero value for the map's value type.
    D. It throws an exception.
    E. None of the above

18. (2 points) How can you safely check if a key exists in a map?
    A. ‘value, exists := m‘
    B. ‘value := m; exists := value != nil‘
    C. ‘exists := m‘
    D. ‘value := m.get(key); exists := value != nil‘
    E. None of the above

19. (2 points) Which statement accurately describes how a type implements an interface in Go?
    A. The type explicitly declares the interface it implements.
    B. The type implicitly implements the interface by defining the required methods.
    C. The type inherits the interface.
    D. The type is annotated with the interface name.
    E. None of the above

20. (2 points) What is an empty interface in Go?
    A. An interface with a single method.
    B. An interface that has been initialized to nil.
    C. An interface that doesn't specify any methods.
    D. An obsolete or deprecated interface.
    E. None of the above

21. (2 points) What does the following code achieve? ‘value, ok := myInterface.(MyType)‘
    A. It asserts that ‘myInterface‘ is of type ‘MyType‘ and retrieves its value.
    B. It converts ‘myInterface‘ to ‘MyType‘ and returns an error if unsuccessful.
    C. It checks if ‘MyType‘ implements ‘myInterface‘.
    D. It creates a new interface named ‘MyType‘ from ‘myInterface‘.
    E. None of the above

22. (2 points) Which of the following can hold any value in Go?
    A. ‘interface‘
    B. ‘interfaceAny‘
    C. ‘type Any interface‘

D. ‘Any‘

E. None of the above

23. (2 points) Which of the following correctly declares a function in Go that takes two integers as arguments and returns an integer?

A. ‘func Sum(int a, int B): int‘

B. ‘int Sum(int a, int B)‘

C. ‘func Sum(a int, b int) int‘

D. ‘Sum(a, b int) -> int‘

E. None of the above

24. (2 points) In Go, how can you return multiple values from a function?

A. By returning an array or slice.

B. By returning a tuple.

C. Using multiple return statements.

D. By specifying multiple return types in the function declaration.

E. None of the above

25. (2 points) What does the ‘defer‘ keyword do in a function?

A. Delays the execution of the function until all other functions are executed.

B. Makes sure the function executes even if the program crashes.

C. Delays the execution of a statement until the surrounding function returns.

D. Delays the execution of the next function in the call stack.

E. None of the above

26. (2 points) What is a closure in Go?

A. A function without a name.

B. A function declared inside another function that captures and can access its outer function’s local variables.

C. A way to close or terminate a function prematurely.

D. A function that returns an error.

E. None of the above

27. (2 points) In Go, how is composition typically achieved?

A. Using class-based inheritance.

B. Embedding one struct type inside another.

C. Implementing multiple interfaces in a single class.

D. Using the ‘extends‘ keyword.

E. None of the above

28. (2 points) Given a ‘Vehicle‘ struct embedded in a ‘Car‘ struct, how can you access a method ‘Drive‘ of the ‘Vehicle‘ struct from a ‘Car‘ instance?

A. ‘Car.Vehicle.Drive()‘
B. ‘Car->Vehicle.Drive()‘
C. ‘Car.Drive()‘
D. ‘Car::Drive()‘
E. None of the above

29. (2 points) Why might you prefer composition over inheritance in Go?
A. Flexibility
B. Reusability
C. Low Coupling
D. Most solutions fit this pattern anyway
E. All of the above

30. (2 points) An "anonymous" struct: is one way to have generic data members for different (but similar) structs and then add more specific items to tailor each of them?
A. True
B. False

31. (2 points) Which keyword is used at the beginning of a Go file to declare its package?
A. ‘import‘
B. ‘module‘
C. ‘package‘
D. ‘use‘
E. None of the above

32. (2 points) What is the significance of the ‘main‘ package in Go?
A. It is used for testing.
B. It indicates the package should be compiled as an executable program.
C. It is the root package for modules.
D. It contains the core libraries of Go.
E. None of the above

33. (2 points) Which file is crucial for defining a module in Go?
A. ‘module.go‘
B. ‘gomodule.txt‘
C. ‘go.pkg‘
D. ‘go.mod‘
E. None of the above

34. (2 points) Which command initializes a new module in Go?
A. ‘go init module‘
B. ‘go new mod‘
C. ‘go mod start‘

D. ‘go mod init‘

E. None of the above

35. (2 points) Which design principle suggests building object capabilities from smaller, reusable pieces instead of strict class hierarchies?

A. Encapsulation Over Polymorphism

B. Segregation Over Integration

C. Inheritance Over Composition

D. Abstraction Over Composition

E. None of the above

36. (2 points) Why might a developer prefer inheritance over composition?

A. Code Reusability.

B. Polymorphism.

C. Structural Hierarchies.

D. The like Java.

E. All of the above

37. (2 points) Which problem is associated with multiple inheritance and can be avoided by using composition?

A. Ellipsis Problem

B. Square Problem

C. Diamond Problem

D. Triangle Issue

E. None of the above

38. (2 points) In Go, how do you define an identifier that can be accessed from outside its package?

A. Prefix it with an underscore (_)

B. It’s determined by the package’s configuration.

C. Start its name with an uppercase letter.

D. End its name with an exclamation mark (!)

E. None of the above

39. (2 points) Which of the following is true about encapsulation in Go?

A. Encapsulation is achieved using ‘defer‘ and ‘type‘ keywords.

B. All methods are publicly accessible regardless of their naming by typing them correctly.

C. Data and methods can be encapsulated by making their identifiers unexported.

D. Encapsulation is not a feature of Go.

E. All of the above

40. (2 points) If you see a function named ‘processData‘ in a Go package, which of the following can you infer?

    A. The function is exported and can be used anywhere.

    B. The function is unexported and can only be used within its package.

    C. The function is unexported but can be accessed using the ‘friend‘ keyword.

    D. The function processes data.

    E. None of the above

41. (2 points) Based on the following snippet, we can without question infer that:

```go
func (w Warrior) Attack() int {
  return w.Strength * 3
}

func (w Warrior) GetName() string {
  return w.c.Name
}
```

    A. There is an anonymous struct defined in Warrior.

    B. There is an empty interface defined.

    C. There is an interface with ‘GetName‘ and ‘Attack‘ defined.

    D. There is nothing we can infer.

    E. None of the above

42. (2 points) Based on the following snippet, we can we infer?

```go
func (p *Person) WithAge(age int) *Person {
    pCopy := *p // Create a copy of the original person
    pCopy.Age = age
    return &pCopy
}
```

    A. This is Go’s version of creating linked nodes.https://www.overleaf.com/project/6514e277500b93

    B. This is Go’s way of return a copy of an existing struct.

    C. There is an interface somewhere that has ‘WithAge‘ in it.

    D. There is nothing we can infer.

    E. None of the above

43. (2 points) What is the significance of the following function?

```go
func MakeSomething() func() int {
  i := 0
  return func() int {
    i++
    return i
  }
}
```

A. It's just a function.

B. It's a function that returns a function.

C. It's Go's version of encapsulation.

D. It's a function that implements a closure.

E. It doesn't work.

44. (2 points) The receiver parameter on a function definition indicates that this function is actually a method for an interface since structs have no methods to define.

A. True

B. False

45. (2 points) Which of the following is False, when it comes to Go?

A. Its a Compiled Language

B. It does Garbage Collection

C. An Imperative Programming Language as well as Procedural

D. Does not support concurrency