**DEPARTMENT OF CSE, CSE(AI&ML) &CSE(DS)**

**GO PROGRAMMING (20-CS-PC-411)**

**Mid-1 Question Bank**

**IV – B. TECH I– SEM**

**UNIT-1**

**PART-A**

1. What is Go programming language? Who developed it and when was it first released?
2. What is the purpose of the main function in Go?
3. How is memory management handled in Go?
4. Describe Go's support for concurrency.
5. How does Go handle garbage collection?
6. How do you read and write data from/to files in Go?
7. What are the advantages of using Go compared to other programming languages?
8. How can you handle command-line arguments in a Go program?
9. How does Go handle errors, and what is the purpose of the error interface?
10. How do you handle exceptions in Go?
11. To set up your machine for Go programming, what is required?
12. Explain three sections in Go program or basic structure of Go program?
13. Why is Go lang fast compared to other languages?
14. What is the role of the GOPATH environment variable?
15. How does the go get command work?
16. Mention the packages in Go Lang?
17. Print Hello World in Go lang?
18. What data types does Go lang use?

**PART-B**

1. What are some of the distinguishing features of the Go programming language that set it apart from other languages such as Python, Java, or C++?
2. Why was Go created? What were some of the problems its creators were trying to solve, and how has it succeeded or failed in solving these problems?
3. Discuss the ways in which Go promotes simplicity and readability in its syntax and how this might be beneficial to a developer or a team of developers.
4. What are the key features of Go's concurrency model? How does it differ from the concurrency models in other languages?
5. How does memory management work in Go? Discuss how the garbage collector operates in the context of Go programming.
6. Discuss the use of Go in modern software development. What types of applications or services might particularly benefit from being written in Go?
7. Go does not support classes in the traditional object-oriented sense. How does it achieve similar functionalities?
8. Discuss the tools and libraries that Go provides for testing and debugging. How do these contribute to the overall developer experience?
9. What does it mean when we say that Go is a statically typed language? What are the pros and cons of this approach?
10. Discuss the Go's standard library. What functionality does it provide out of the box?
11. How does error handling work in Go? Contrast this with how errors are handled in other programming languages.
12. Describe the ecosystem surrounding Go. What sorts of resources (e.g., package managers, libraries, community support, etc.) are available to Go developers?
13. What are some of the criticisms of Go, and how have the language's developers and community responded to these criticisms?
14. How can you concatenate string values? What happens when concatenating strings? Explain with an example.
15. Explain why concurrency is not parallelism?
16. What is garbage collection, type safety in Golang?

**UNIT-2**

1. How is a variable declared in Go?
2. What are the different types of variables that can be declared in Go?
3. How do you declare multiple variables at once in Go?
4. What does it mean when a variable is declared as a constant in Go? Can its value be changed?
5. What is type inference in Go? How is it used when declaring variables?
6. What is the scope of a variable in Go? How does it change when the variable is declared inside a function vs outside?
7. What does it mean when a variable is declared as exported in Go?
8. How do you declare and use a global variable in Go?
9. What are the naming conventions for variables in Go?
10. What is the difference between a local and a global variable in Go?
11. How do you declare and use an array, slice, map struct variable in Go?
12. How can you convert a variable of one type to another type in Go?
13. How can you use a variable to store the result of a function that returns multiple values in Go?
14. Explain the purpose and usage of blank identifier (\_) in variable declaration.
15. How is string concatenation performed in Go?
16. How are Boolean variables declared and used in Go?
17. How to create and use an enumerated constant (iota) in Go?
18. What is the role of package OS in handling environment variables in Go?
19. Explain the concept of type inference in Go. How is it used when declaring variables?
20. What is the scope of a variable in Go? How does it change when the variable is declared inside a function versus outside?
21. How are variables initialized in Go? Are there default values for different types?
22. What is the difference between short variable declaration and regular variable declaration in Go?
23. How do you declare and use constants in Go?
24. How can you declare multiple variables at once in Go?
25. Describe the naming conventions for variables in Go.
26. How can you convert a variable of one type to another type in Go?
27. What is the difference between local and global variables in Go?
28. How are pointers used in Go variables? Provide an example.
29. How can you use variables to store the results of functions that return multiple values in Go?
30. How are arrays and slices used to store multiple values in Go variables?
31. What are the key differences between arrays and slices in Go?
32. What are the advantages of using maps over other data structures in Go?
33. How can you delete elements from a map in Go?

**UNIT-3**

1. How do you declare a function in Go? Provide an example.
2. How do you pass arguments to a function in Go? Are arguments passed by value or reference?
3. What are the differences between a named return value and a regular return value in Go functions?
4. How do you create a pointer variable in Go? What are its uses?
5. Explain the use of variadic functions in Go. Provide an example.
6. How can you return multiple values from a single function in Go?
7. How do you create anonymous functions (closures) in Go? Provide an example.
8. What is a callback function, and how can it be implemented in Go?
9. How can you use function types and function values in Go?
10. How do you handle errors in functions in Go?
11. What is a recursive function in Go? Provide an example.
12. How do you pass a function as an argument to another function in Go?
13. Explain the concept of method receivers in Go functions.
14. How do you define a function that takes multiple arguments in Go?
15. How do you define a function that returns multiple values in Go?
16. What is a variadic function in Go? Give an example.
17. How are deferred function calls used in Go?
18. What is a function signature in Go?
19. How do you return multiple values from a function in Go?
20. How are parameters passed to functions in Go, by value or by reference?
21. What is a function closure in Go?
22. How is error handling typically done in Go functions?
23. How are package names related to folder structure in Go?
24. What is the purpose of the main package and main function in Go?
25. Describe the use of the fmt and os packages in Go.
26. How do you handle package level variables?
27. What is a Go module and how does it relate to packages?
28. Describe the use of the init, net/http package in Go.
29. What is the significance of the main package in Go?
30. What is the difference between exported and unexported identifiers in Go packages?
31. How do you document a package, function, or type in Go?
32. How do you manage package dependencies in Go?
33. How do you create custom packages in Go?
34. How does defer, panic and recover statements work in Go functions?