Slip 1 A) Write a program in GO language to accept user choice and print answers using arithmetic operators.

Ans.

```go
package main

import (
    "fmt"
)

func main() {
    var num1, num2 float64
    var operator string

    // prompt the user to enter two numbers
    fmt.Print("Enter first number: ")
    fmt.Scanln(&num1)
    fmt.Print("Enter second number: ")
    fmt.Scanln(&num2)

    // prompt the user to enter an operator
    fmt.Print("Enter operator (+, -, *, /): ")
    fmt.Scanln(&operator)

    var result float64
    switch operator {
    case "+":
        result = num1 + num2
    case "-":
        result = num1 - num2
    case "*":
        result = num1 * num2
    case "/":
        result = num1 / num2
    default:
        fmt.Println("Invalid operator entered.")
        return
    }

    // display the result
    fmt.Printf("%.2f %s %.2f = %.2f\n", num1, operator, num2, result)
}
```

Slip 1 b) Write a program in GO language to accept n student details like roll_no, stud_name, mark1,mark2, mark3. Calculate the total and average of marks using structure.

```
package main

import "fmt"

type student struct {
    rollNo  int
    name    string
    mark1   float64
    mark2   float64
    mark3   float64
    total   float64
    average float64
}

func main() {
    var n int
    fmt.Print("Enter the number of students: ")
    fmt.Scanln(&n)

    students := make([]student, n)

    for i := 0; i < n; i++ {
        fmt.Printf("Enter details for student %d:\n", i+1)
        fmt.Print("Enter roll number: ")
        fmt.Scanln(&students[i].rollNo)
        fmt.Print("Enter name: ")
        fmt.Scanln(&students[i].name)
        fmt.Print("Enter marks for three subjects: ")
        fmt.Scanln(&students[i].mark1, &students[i].mark2, &students[i].mark3)

        students[i].total = students[i].mark1 + students[i].mark2 +
students[i].mark3
        students[i].average = students[i].total / 3
    }

    fmt.Println("Total and average marks for each student:")
    for _, s := range students {
        fmt.Printf("%s (Roll No. %d): Total = %.2f, Average = %.2f\n", s.name,
s.rollNo, s.total, s.average)
    }
}
```

Slip 2 A. Write a program in GO language to print Fibonacci series of n terms.

```go
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Enter the number of terms: ")
    fmt.Scanln(&n)

    // Initialize the first two terms of the series
    a, b := 0, 1

    // Print the first n terms of the series
    fmt.Printf("Fibonacci series of %d terms:\n", n)
    for i := 0; i < n; i++ {
        fmt.Print(a, " ")

        // Calculate the next term of the series
        a, b = b, a+b
    }
}
```

Slip 2 B. Write a program in GO language to print file information.

```go
package main

import (
    "fmt"
    "os"
)

func main() {
    // Prompt the user to enter the file name
    var fileName string
    fmt.Print("Enter the file name: ")
    fmt.Scanln(&fileName)

    // Open the file and get its information
    fileInfo, err := os.Stat(fileName)
```

```
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    // Print the file information
    fmt.Println("File name:", fileInfo.Name())
    fmt.Println("File size (bytes):", fileInfo.Size())
    fmt.Println("File mode:", fileInfo.Mode())
    fmt.Println("Is directory?", fileInfo.IsDir())
    fmt.Println("File modification time:", fileInfo.ModTime())
}
```

Slip 3 A. Write a program in the GO language using function to check whether accepts number is palindrome or not.

```
package main

import "fmt"

func main() {

    var palNum, remainder int

    fmt.Print("Enter the Number to check Palindrome = ")
    fmt.Scanln(&palNum)

    reverse := 0

    for temp := palNum; temp > 0; temp = temp / 10 {
        remainder = temp % 10
        reverse = reverse*10 + remainder
    }

    fmt.Println("The Reverse of the Given Number = ", reverse)
    if palNum == reverse {
        fmt.Println(palNum, " is a Palindrome Number")
    } else {
        fmt.Println(palNum, " is Not a Palindrome Number")
    }
}
```

Slip 3 B. Write a Program in GO language to accept n records of employee information (eno,ename,salary) and display record of employees having maximum salary.

```go
package main

import (
    "fmt"
)

func main() {
    var n int
    fmt.Print("Enter the number of employees: ")
    fmt.Scan(&n)

    var eno int
    var ename string
    var salary float64
    maxSalary := 0.0

    for i := 1; i <= n; i++ {
        fmt.Printf("Enter details of employee %d:\n", i)
        fmt.Print("Employee number: ")
        fmt.Scan(&eno)
        fmt.Print("Employee name: ")
        fmt.Scan(&ename)
        fmt.Print("Employee salary: ")
        fmt.Scan(&salary)

        if salary > maxSalary {
            maxSalary = salary
        }
    }

    fmt.Println("\nEmployees with maximum salary:")
    for i := 1; i <= n; i++ {
        fmt.Printf("Enter details of employee %d:\n", i)
        fmt.Print("Employee number: ")
        fmt.Scan(&eno)
        fmt.Print("Employee name: ")
        fmt.Scan(&ename)
        fmt.Print("Employee salary: ")
        fmt.Scan(&salary)

        if salary == maxSalary {
```

```
            fmt.Printf("Employee number: %d\n", eno)
            fmt.Printf("Employee name: %s\n", ename)
            fmt.Printf("Employee salary: %.2f\n", salary)
        }
    }
}
```

Slip 4 A. Write a program in GO language to print a recursive sum of digits of a given number.

```
package main

import (
    "fmt"
)

func recursiveSum(num int) int {
    if num < 10 {
        return num
    }
    return recursiveSum(num/10) + num%10
}

func main() {
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scan(&num)

    sum := recursiveSum(num)

    fmt.Printf("The recursive sum of digits of %d is %d\n", num, sum)
}
```

Slip 4 B. Write a program in GO language to sort array elements in ascending order.

```
package main

import (
    "fmt"
    "sort"
)
```

```
func main() {
    var n int
    fmt.Print("Enter the size of the array: ")
    fmt.Scan(&n)

    arr := make([]int, n)
    for i := 0; i < n; i++ {
        fmt.Printf("Enter element %d: ", i+1)
        fmt.Scan(&arr[i])
    }

    sort.Ints(arr)

    fmt.Println("Sorted array in ascending order:", arr)
}
```

Slip 5 A. Write a program in GO language program to create Text file.

```
package main

import (
    "fmt"
    "os"
)

func main() {
    file, err := os.Create("myfile.txt")
    if err != nil {
        fmt.Println("Error creating file:", err)
        return
    }
    defer file.Close()

    fmt.Fprintln(file, "This is a sample text file created using GO programming
language!")
    fmt.Fprintln(file, "We can write any text here using the 'fmt' package and
the 'Fprintln' function.")
    fmt.Fprintln(file, "This text will be written to the file in separate
lines.")

    fmt.Println("Text file created successfully!")
}
```

Slip 5 B. Write a program in GO language to accept n records of employee information (eno,ename,salary) and display records of employees having minimum salary.

```
package main

import (
    "fmt"
)

type Employee struct {
    eno    int
    ename  string
    salary int
}

func main() {
    var n int
    fmt.Print("Enter the number of employees: ")
    fmt.Scan(&n)

    employees := make([]Employee, n)

    for i := 0; i < n; i++ {
        fmt.Printf("\nEnter the details of employee %d:\n", i+1)
        fmt.Print("Enter employee number: ")
        fmt.Scan(&employees[i].eno)
        fmt.Print("Enter employee name: ")
        fmt.Scan(&employees[i].ename)
        fmt.Print("Enter employee salary: ")
        fmt.Scan(&employees[i].salary)
    }

    minSalary := employees[0].salary
    for i := 0; i < n; i++ {
        if employees[i].salary < minSalary {
            minSalary = employees[i].salary
        }
    }

    fmt.Printf("\nEmployees with minimum salary of %d:\n", minSalary)
    for i := 0; i < n; i++ {
        if employees[i].salary == minSalary {
            fmt.Printf("Employee %d - Name: %s, Salary: %d\n", employees[i].eno,
employees[i].ename, employees[i].salary)
```

```
        }
    }
}
```

Slip 6 A. Write a program in GO language to accept two matrices and display its multiplication.

```
package main

import (
    "fmt"
)

func main() {
    var rowsA, colsA, rowsB, colsB int

    fmt.Print("Enter the number of rows of matrix A: ")
    fmt.Scan(&rowsA)
    fmt.Print("Enter the number of columns of matrix A: ")
    fmt.Scan(&colsA)

    fmt.Print("Enter the number of rows of matrix B: ")
    fmt.Scan(&rowsB)
    fmt.Print("Enter the number of columns of matrix B: ")
    fmt.Scan(&colsB)

    if colsA != rowsB {
        fmt.Println("Matrices cannot be multiplied.")
        return
    }

    // Accept matrix A
    fmt.Println("Enter the elements of matrix A:")
    matrixA := make([][]int, rowsA)
    for i := range matrixA {
        matrixA[i] = make([]int, colsA)
        for j := range matrixA[i] {
            fmt.Scan(&matrixA[i][j])
        }
    }

    // Accept matrix B
    fmt.Println("Enter the elements of matrix B:")
    matrixB := make([][]int, rowsB)
```

```
    for i := range matrixB {
        matrixB[i] = make([]int, colsB)
        for j := range matrixB[i] {
            fmt.Scan(&matrixB[i][j])
        }
    }

    // Multiply matrices
    result := make([][]int, rowsA)
    for i := range result {
        result[i] = make([]int, colsB)
        for j := range result[i] {
            for k := 0; k < colsA; k++ {
                result[i][j] += matrixA[i][k] * matrixB[k][j]
            }
        }
    }

    // Print result
    fmt.Println("Resultant matrix:")
    for i := range result {
        for j := range result[i] {
            fmt.Printf("%d ", result[i][j])
        }
        fmt.Println()
    }
}
```

Slip 6 B. Write a program in GO language to copy all elements of one array into another using a method.

```
package main

import "fmt"

func main() {
    // create the source array
    source := []int{1, 2, 3, 4, 5}

    // create a destination array with the same size as source
    destination := make([]int, len(source))

    // call the copyArray method to copy elements
    copyArray(source, destination)
```

```
    // print the contents of destination array
    fmt.Println("Contents of destination array:")
    fmt.Println(destination)
}

// copyArray method to copy elements from source to destination
func copyArray(source []int, destination []int) {
    for i := range source {
        destination[i] = source[i]
    }
}
```

Slip 7 A. Write a program in GO language to accept one matrix and display its transpose.

```
package main

import "fmt"

func main() {
    var rows, cols int

    fmt.Print("Enter the number of rows of matrix: ")
    fmt.Scan(&rows)
    fmt.Print("Enter the number of columns of matrix: ")
    fmt.Scan(&cols)

    // Accept matrix
    fmt.Println("Enter the elements of matrix:")
    matrix := make([][]int, rows)
    for i := range matrix {
        matrix[i] = make([]int, cols)
        for j := range matrix[i] {
            fmt.Scan(&matrix[i][j])
        }
    }

    // Print original matrix
    fmt.Println("Original matrix:")
    for i := range matrix {
        for j := range matrix[i] {
            fmt.Printf("%d ", matrix[i][j])
        }
```

```
        fmt.Println()
    }

    // Transpose matrix
    transpose := make([][]int, cols)
    for i := range transpose {
        transpose[i] = make([]int, rows)
        for j := range transpose[i] {
            transpose[i][j] = matrix[j][i]
        }
    }

    // Print transpose matrix
    fmt.Println("Transpose of matrix:")
    for i := range transpose {
        for j := range transpose[i] {
            fmt.Printf("%d ", transpose[i][j])
        }
        fmt.Println()
    }
}
```

Slip 7 B. Write a program in GO language to create structure student. Write a method show() whose receiver is a pointer of struct student.

```
package main

import "fmt"

type Student struct {
    Name string
    Age  int
}

func (s *Student) show() {
    fmt.Printf("Name: %s, Age: %d\n", s.Name, s.Age)
}

func main() {
    // Create a student object
    student := Student{Name: "John Doe", Age: 20}
```

```
    // Call the show() method on the student object using a pointer receiver
    (&student).show() // or simply, student.show()
}
```

Slip 8 A. Write a program in GO language to accept the book details such as BookID, Title, Author, Price. Read and display the details of ‘n’ number of books.

```
package main

import "fmt"

type Book struct {
    BookID int
    Title  string
    Author string
    Price  float64
}

func main() {
    var n int
    fmt.Print("Enter the number of books: ")
    fmt.Scan(&n)

    // Create a slice of n books
    books := make([]Book, n)

    // Accept book details from user
    for i := 0; i < n; i++ {
        fmt.Printf("Enter the details of book %d:\n", i+1)
        fmt.Print("Book ID: ")
        fmt.Scan(&books[i].BookID)
        fmt.Print("Title: ")
        fmt.Scan(&books[i].Title)
        fmt.Print("Author: ")
        fmt.Scan(&books[i].Author)
        fmt.Print("Price: ")
        fmt.Scan(&books[i].Price)
        fmt.Println()
    }

    // Display book details to user
    fmt.Println("Book details:")
```

```
    for i, book := range books {
        fmt.Printf("Book %d:\n", i+1)
        fmt.Printf("Book ID: %d\n", book.BookID)
        fmt.Printf("Title: %s\n", book.Title)
        fmt.Printf("Author: %s\n", book.Author)
        fmt.Printf("Price: %.2f\n", book.Price)
        fmt.Println()
    }
}
```

Slip 8 B. Write a program in GO language to create an interface shape that includes area and perimeter. Implements these methods in circle and rectangle type.

```
package main

import (
    "fmt"
    "math"
)

type Shape interface {
    area() float64
    perimeter() float64
}

type Circle struct {
    radius float64
}

func (c Circle) area() float64 {
    return math.Pi * c.radius * c.radius
}

func (c Circle) perimeter() float64 {
    return 2 * math.Pi * c.radius
}

type Rectangle struct {
    length float64
    width  float64
}
```

```go
func (r Rectangle) area() float64 {
    return r.length * r.width
}

func (r Rectangle) perimeter() float64 {
    return 2 * (r.length + r.width)
}

func main() {
    c := Circle{radius: 5}
    r := Rectangle{length: 4, width: 6}

    fmt.Printf("Area of circle: %.2f\n", c.area())
    fmt.Printf("Perimeter of circle: %.2f\n", c.perimeter())
    fmt.Printf("Area of rectangle: %.2f\n", r.area())
    fmt.Printf("Perimeter of rectangle: %.2f\n", r.perimeter())
}
```

Slip 9 A Write a program in GO language using a function to check whether the accepted number is palindrome or not.

```go
package main

import (
    "fmt"
    "strconv"
)

func isPalindrome(num int) bool {
    // Convert the number to a string
    str := strconv.Itoa(num)

    // Check if the string is equal to its reverse
    for i := 0; i < len(str)/2; i++ {
        if str[i] != str[len(str)-1-i] {
            return false
        }
    }
    return true
}

func main() {
```

```
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scanln(&num)

    if isPalindrome(num) {
        fmt.Println(num, "is a palindrome")
    } else {
        fmt.Println(num, "is not a palindrome")
    }
}
```

Slip 9 B Write a program in GO language to create an interface shape that includes area and volume. Implements these methods in square and rectangle type.

```
package main
import (
    "fmt"
)
type shape interface {
    area() float64
    volume() float64
}
type square struct {
    length float64
}
func (s square) area() float64 {
    return s.length * s.length
}
func (s square) volume() float64 {
    return 0
}
type rectangle struct {
    length float64
    width  float64
    height float64
}
func (r rectangle) area() float64 {
    return r.length * r.width
}
func (r rectangle) volume() float64 {
    return r.length * r.width * r.height
}
func main() {
    shapes := []shape{square{5}, rectangle{4, 3, 2}}
```

```
    for _, s := range shapes {
        fmt.Println("Area:", s.area())
        fmt.Println("Volume:", s.volume())
    }
}
```

Slip 10 A Write a program in GO language to create an interface and display its values with the help of type assertion.

```
package main
import "fmt"
type geometry interface {
    area() float64
    perimeter() float64
}
type rectangle struct {
    width, height float64
}
func (r rectangle) area() float64 {
    return r.width * r.height
}
func (r rectangle) perimeter() float64 {
    return 2*r.width + 2*r.height
}
func main() {
    var g geometry = rectangle{width: 3, height: 4}
    fmt.Println(g)
    if r, ok := g.(rectangle); ok {
        fmt.Println("Width:", r.width)
        fmt.Println("Height:", r.height)
        fmt.Println("Area:", r.area())
        fmt.Println("Perimeter:", r.perimeter())
    } else {
        fmt.Println("Not a rectangle")
    }
}
```

Slip 10 B Write a program in GO language to read and write Fibonacci series to the using channel. It not in the slip for exam.

Slip 11 A. Write a program in GO language to check whether the accepted number is two digit or not.

```go
package main

import "fmt"

func main() {
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scanln(&num)
    if num >= 10 && num <= 99 {
        fmt.Println("The number is two digits")
    } else {
        fmt.Println("The number is not two digits")
    }
}
```

Slip 11 B. Write a program in GO language to create a buffered channel, store few values in it and find channel capacity and length. Read values from channel and find modified length of a channel. Not coming in the slip for exam.

Slip 12 A. Write a program in GO language to swap two numbers using call by reference concept.

```go
package main

import "fmt"

func swap(a *int, b *int) {
    temp := *a
    *a = *b
    *b = temp
}

func main() {
    var num1, num2 int

    fmt.Print("Enter the first number: ")
```

```
    fmt.Scanln(&num1)

    fmt.Print("Enter the second number: ")
    fmt.Scanln(&num2)

    fmt.Printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2)

    swap(&num1, &num2)

    fmt.Printf("After swapping: num1 = %d, num2 = %d\n", num1, num2)
}
```

Slip 12 B. Write a program in GO language that creates a slice of integers, checks numbers from the slice are even or odd and further sent to respective go routines through channel and display values received by goroutines. Not coming in exam.

Slip 13 A. Write a program in GO language to print sum of all even and odd numbers separately between 1 to 100.

```
package main

import "fmt"

func main() {
    evenSum := 0
    oddSum := 0

    for i := 1; i <= 100; i++ {
        if i%2 == 0 {
            evenSum += i
        } else {
            oddSum += i
        }
    }

    fmt.Printf("Sum of all even numbers between 1 to 100 is %d\n", evenSum)
    fmt.Printf("Sum of all odd numbers between 1 to 100 is %d\n", oddSum)
}
```

Slip 13 B. Write a function in GO language to find the square of a number and write a benchmark for it. Not coming in exam.

Slip 14 A. Write a program in GO language to demonstrate working of slices (like append, remove, copy etc.)

```go
package main

import "fmt"

func main() {
    // Creating a slice
    nums := []int{1, 2, 3, 4, 5}
    fmt.Println("Original slice:", nums)

    // Appending elements to the slice
    nums = append(nums, 6, 7, 8)
    fmt.Println("Slice after appending:", nums)

    // Removing an element from the slice
    nums = append(nums[:3], nums[4:]...)
    fmt.Println("Slice after removing 4th element:", nums)

    // Copying a slice
    newNums := make([]int, len(nums))
    copy(newNums, nums)
    fmt.Println("New slice after copying:", newNums)
}
```

Slip 14 B. Write a program in GO language using go routine and channel that will print the sum of the squares and cubes of the individual digits of a number. Example if number is 123 then
squares = (1 * 1) + (2 * 2) + (3 * 3)
cubes = (1 * 1 * 1) + (2 * 2 * 2) + (3 * 3 * 3).

```go
package main

import "fmt"

func main() {
```

```
    // Creating a slice
    nums := []int{1, 2, 3, 4, 5}
    fmt.Println("Original slice:", nums)

    // Appending elements to the slice
    nums = append(nums, 6, 7, 8)
    fmt.Println("Slice after appending:", nums)

    // Removing an element from the slice
    nums = append(nums[:3], nums[4:]...)
    fmt.Println("Slice after removing 4th element:", nums)

    // Copying a slice
    newNums := make([]int, len(nums))
    copy(newNums, nums)
    fmt.Println("New slice after copying:", newNums)
}
```

Slip 15 A. Write a program in GO language to demonstrate function return multiple values.

```
package main

import "fmt"

// Function to calculate sum and product of two numbers
func calcSumAndProduct(a, b int) (int, int) {
    sum := a + b
    product := a * b
    return sum, product
}

func main() {
    // Calling the calcSumAndProduct() function and storing the results in two
variables
    sum, product := calcSumAndProduct(5, 10)

    fmt.Printf("Sum: %d, Product: %d\n", sum, product)
}
```

Slip 15 B. Write a program in GO language to read XML file into structure and display structure.

```go
package main

import (
    "encoding/xml"
    "fmt"
    "os"
)

type Book struct {
    Title       string `xml:"title"`
    Author      string `xml:"author"`
    Publication string `xml:"publication"`
    Pages       int    `xml:"pages"`
}

func main() {
    file, err := os.Open("books.xml")
    if err != nil {
        fmt.Println("Error opening file:", err)
        return
    }
    defer file.Close()
    decoder := xml.NewDecoder(file)
    var books []Book
    for {
        token, err := decoder.Token()
        if err != nil {
            break
        }
        if elem, ok := token.(xml.StartElement); ok && elem.Name.Local == "book"
{
            var book Book
            decoder.DecodeElement(&book, &elem)
            books = append(books, book)
        }
    }
    for _, book := range books {
        fmt.Printf("Title: %s\nAuthor: %s\nPublication: %s\nPages: %d\n\n",
            book.Title, book.Author, book.Publication, book.Pages)
    }
}
```

Slip 16 A. Write a program in GO language to create a user defined package to find out the area of a rectangle.

```
// First, create a new folder for the package called rectangle with a file named
rectangle.go.
//  Here's the code for rectangle.go:


package rectangle

func Area(length, width float64) float64 {
    return length * width
}

// This package contains a single function Area() that calculates the area of a
rectangle with the given length and width.

// Next, create a main program that uses this package to calculate the area of a
rectangle. Here's an example program:

package main

import (
    "fmt"
    "rectangle"
)

func main() {
    length := 10.0
    width := 5.0
    area := rectangle.Area(length, width)
    fmt.Printf("The area of the rectangle with length %f and width %f is %f",
length, width, area)
}

// In this program, we import the rectangle package and use the Area() function
to calculate the area of a rectangle with length 10 and
// width 5. Finally, we print the result using fmt.Printf().
```

Slip 16 B. Write a program in GO language that prints out the numbers from 0 to 10, waiting between 0 and 250 ms after each one using the delay function.

```go
package main

import (
    "fmt"
    "time"
)

func main() {
    for i := 0; i <= 10; i++ {
        fmt.Println(i)
        time.Sleep(time.Duration(250) * time.Millisecond)
    }
}
```

Slip 17 A. Write a program in GO language to illustrate the concept of returning multiple values from a function. (Add, Subtract, Multiply, Divide).

```go
package main

import "fmt"

func AddSubtractMultiplyDivide(a, b float64) (float64, float64, float64, float64)
{
    return a + b, a - b, a * b, a / b
}

func main() {
    x := 10.0
    y := 5.0
    add, subtract, multiply, divide := AddSubtractMultiplyDivide(x, y)
    fmt.Printf("Addition of %.2f and %.2f is %.2f\n", x, y, add)
    fmt.Printf("Subtraction of %.2f from %.2f is %.2f\n", y, x, subtract)
    fmt.Printf("Multiplication of %.2f and %.2f is %.2f\n", x, y, multiply)
    fmt.Printf("Division of %.2f by %.2f is %.2f\n", x, y, divide)
}
```

Slip 17 B. Write a program in GO language to add or append content at the end of a text file.

```go
package main

import (
    "fmt"
    "os"
)

func main() {
    // Open file for appending content
    file, err := os.OpenFile("myfile.txt", os.O_WRONLY|os.O_APPEND|os.O_CREATE,
0644)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer file.Close()

    // Add content to file
    content := "\nThis is the new content to add to the file"
    _, err = file.WriteString(content)
    if err != nil {
        fmt.Println(err)
        return
    }

    fmt.Println("Content added successfully!")
}
```

Slip 18 A. Write a program in GO language to print a multiplication table of number using function.

```go
package main

import "fmt"

func printMultiplicationTable(num int) {
    for i := 1; i <= 10; i++ {
        fmt.Printf("%d x %d = %d\n", num, i, num*i)
    }
}

func main() {
    var num int
```

```
    fmt.Println("Enter a number to print its multiplication table:")
    fmt.Scanln(&num)
    printMultiplicationTable(num)
}
```

Slip 18 B. Write a program in GO language using a user defined package calculator that performs one calculator operation as per the user's choice.

```
package main

import (
    "calculator"
    "fmt"
)

func main() {
    var num1, num2 float64
    var choice int

    fmt.Println("Enter two numbers:")
    fmt.Scanln(&num1, &num2)

    fmt.Println("Enter the operation you want to perform:")
    fmt.Println("1. Addition")
    fmt.Println("2. Subtraction")
    fmt.Println("3. Multiplication")
    fmt.Println("4. Division")
    fmt.Scanln(&choice)

    switch choice {
    case 1:
        result := calculator.Add(num1, num2)
        fmt.Printf("Result: %.2f", result)
    case 2:
        result := calculator.Subtract(num1, num2)
        fmt.Printf("Result: %.2f", result)
    case 3:
        result := calculator.Multiply(num1, num2)
        fmt.Printf("Result: %.2f", result)
    case 4:
        result, err := calculator.Divide(num1, num2)
        if err != nil {
            fmt.Println(err)
```

```
        } else {
            fmt.Printf("Result: %.2f", result)
        }
    default:
        fmt.Println("Invalid choice!")
    }
}
```

Slip 19 A. Write a program in GO language to illustrate the function returning multiple values(add, subtract).

```
package main

import "fmt"

// add function returns sum of two integers
func add(a, b int) int {
    return a + b
}

// subtract function returns difference between two integers
func subtract(a, b int) int {
    return a - b
}

func main() {
    // calling add function and printing returned value
    sum := add(10, 20)
    fmt.Println("Sum:", sum)

    // calling subtract function and printing returned value
    diff := subtract(20, 10)
    fmt.Println("Difference:", diff)
}
```

Slip 19 B. Write a program in the GO language program to open a file in READ only mode.

```
package main

import (
    "fmt"
```

```
    "os"
)

func main() {
    // Opening a file in read-only mode
    file, err := os.Open("example.txt")
    defer file.Close()

    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    // Reading contents of the file
    data := make([]byte, 100)
    count, err := file.Read(data)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    // Printing the read data
    fmt.Printf("Read %d bytes: %s\n", count, data[:count])
}
```

Slip 20 A. Write a program in Go language to add or append content at the end of a text file.

```
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    // Opening the file in append mode
    file, err := os.OpenFile("example.txt", os.O_APPEND|os.O_WRONLY, 0644)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }
```

```
    defer file.Close()

    // Getting the input text to append
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Enter the text to append: ")
    scanner.Scan()
    inputText := scanner.Text()

    // Appending the input text to the file
    _, err = file.WriteString(inputText + "\n")
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    fmt.Println("Content added to the file.")
}
```

Slip 20 B. Write a program in Go language how to create a channel and illustrate how to close a channel using for range loop and close function. Not coming for exam.