A1_A1
Wap in go language to print student name, rollno, division and college name

```
package main

import "fmt"

// Declaration of structure
type Student struct {
	Id       int
	Name     string
	division string
	clg      string
}

func main() {
	stu1 := Student{Id: 101, Name: "Kapil", division: "A", clg: "DYP"}
	stu2 := Student{Id: 102, Name: "Amit", division: "B", clg: "DYP"}
	stu3 := Student{Id: 103, Name: "Arun", division: "C", clg: "DYP"}

	fmt.Printf("Student Infomation:")
	fmt.Println("\nStudent1: ", stu1)
	fmt.Println("\nStudent2: ", stu2)
	fmt.Println("\nStudent3: ", stu3)
}
```

A1_A2
Wap in go language to print whether number is even or odd

```
package main

import "fmt"

func main() {
	var num int
	num = 5
	if num%2 == 0 {
		fmt.Printf("num is EVEN")
	} else {
		fmt.Printf("num is ODD")
	}
}
```

A1_A3
Wap in go language to swap the number without temporary variable

```
package main

import "fmt"
```

```
func swap(a, b int) {
	fmt.Printf("Before swapping, numbers are %d and %d\n", a, b)
	b = a + b
	a = b - a
	b = b - a
	fmt.Printf("After swapping, numbers are %d and %d\n", a, b)
}

func main() {
	swap(20, 40)
	swap(50, 100)
}
```

A1_A4

Wap in go language to print address of a variable

```
package main

import "fmt"

type A struct {
	Number int
	Text   string
}

func main() {
	// array
	arr := [3]int{1, 2, 3}
	fmt.Printf("Address of array = %v: %p\n", arr, &arr)

	// slice
	slice := []int{1, 2, 3}
	fmt.Printf("Address of slice = %v: %p\n", slice, &slice)

	// struct
	structInstance := A{Number: 23, Text: "abc"}
	fmt.Printf("Address of struct = %+v: %p\n", structInstance,
&structInstance)

	// struct field
	fmt.Printf("Address of struct field = %s: %p\n",
structInstance.Text, &structInstance.Text)

	// map
	mapInstance := map[int]int{
		0: 1,
	}
	fmt.Printf("Address of map = %v: %p\n", mapInstance, &mapInstance)
}
```

A1_B1

Wap in go language to print table of given number

```go
package main

import "fmt"

func main() {
	var num int = 0

	fmt.Print("Enter Number: ")
	fmt.Scanf("%d", &num)

	for count := 1; count <= 10; count++ {
		fmt.Printf("%d * %d = %d\n", num, count, num*count)
	}
}
```

A1_B2

Wap in go language to print PASCALS triangle

```go
package main

import "fmt"

func main() {
	var rows int
	var temp int = 1
	fmt.Print("Enter number of rows : ")
	fmt.Scan(&rows)

	for i := 0; i < rows; i++ {

		for j := 1; j <= rows-i; j++ {
			fmt.Print(" ")
		}

		for k := 0; k <= i; k++ {

			if k == 0 || i == 0 {
				temp = 1
			} else {
				temp = temp * (i - k + 1) / k
			}

			fmt.Printf(" %d", temp)
		}
		fmt.Println("")

	}
}
```

A1_B3

Wap in go language to print Fibonacci series of n terms

```
package main

import "fmt"

func main() {
	var n int
	t1 := 0
	t2 := 1
	nextTerm := 0

	fmt.Print("Enter the number of terms : ")
	fmt.Scan(&n)
	fmt.Print("Fibonacci Series :")
	for i := 1; i <= n; i++ {
		if i == 1 {
			fmt.Print(" ", t1)
			continue
		}
		if i == 2 {
			fmt.Print(" ", t2)
			continue
		}
		nextTerm = t1 + t2
		t1 = t2
		t2 = nextTerm
		fmt.Print(" ", nextTerm)
	}
}
```

A1_B4

Wap in go language to illustrate pointer to pointer concept

```
package main

import "fmt"

// Main Function
func main() {

	// taking a variable
	// of integer type
	var v int = 100

	// taking a pointer
	// of integer type
	var pt1 *int = &v
```

```
        // taking pointer to
        // pointer to pt1
        // storing the address
        // of pt1 into pt2
        var pt2 **int = &pt1

        fmt.Println("The Value of Variable v is = ", v)

        // changing the value of v by assigning
        // the new value to the pointer pt1
        *pt1 = 200

        fmt.Println("Value stored in v after changing pt1 = ", v)

        // changing the value of v by assigning
        // the new value to the pointer pt2
        **pt2 = 300

        fmt.Println("Value stored in v after changing pt2 = ", v)
}
```

A1_B5

Wap in go language to explain new function

```
package main

import "fmt"

type sum struct {
        x_val int
        y_val int
}

func main() {
        var p1 = new(int)

        fmt.Println(p1)

        var p = new(sum)

        p.x_val = 10
        p.y_val = 20
        fmt.Println(p)
}
```

A1_C1

WAP in go language to concatenate two strings using pointers.

```
package main

import "fmt"
```

```
func main() {
	var string1 string
	var string2 string
	var str1 *string
	var str2 *string

	fmt.Println("first string")
	fmt.Scanln(&string1)
	fmt.Println("second string")
	fmt.Scanln(&string2)

	str1 = &string1
	str2 = &string2
	fmt.Println("string1:", *str1)
	fmt.Println("string2:", *str2)

	*str1 += *str2

	fmt.Println("the concatenated string is %s", *str1)
}
```

A1_C2

WAP in go language to accept two strings and compare them.

```
package main

import (
	"fmt"
)

func main() {
	var str1, str2 string
	fmt.Print("Enter the first string: ")
	fmt.Scan(&str1)
	fmt.Print("Enter the second string: ")
	fmt.Scan(&str2)

	if str1 == str2 {
		fmt.Println("The two strings are equal.")
	} else {
		fmt.Println("The two strings are not equal.")
	}
}
```

A1_C3

WAP in go language to accept user choice and print answer of using arithmetical

```
package main

import (
	"fmt"
)
```

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

	fmt.Print("Enter the first number: ")
	fmt.Scan(&num1)
	fmt.Print("Enter the second number: ")
	fmt.Scan(&num2)
	fmt.Print("Enter the operator (+, -, *, /): ")
	fmt.Scan(&operator)

	switch operator {
	case "+":
		fmt.Println(num1 + num2)
	case "-":
		fmt.Println(num1 - num2)
	case "*":
		fmt.Println(num1 * num2)
	case "/":
		fmt.Println(num1 / num2)
	default:
		fmt.Println("Invalid operator")
	}
}
```

A1_C4

WAP in go language to check whether accepted number is single digit or not.

```
package main

import (
	"fmt"
)

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

	if num >= 0 && num <= 9 {
		fmt.Println("The number is a single digit.")
	} else {
		fmt.Println("The number is not a single digit.")
	}
}
```

A1_C5

WAP in go language to check whether first string is substring of another string or not.

```
package main

import (
        "fmt"
        "strings"
)

func main() {
        var str1, str2 string
        fmt.Print("Enter the first string: ")
        fmt.Scan(&str1)
        fmt.Print("Enter the second string: ")
        fmt.Scan(&str2)

        if strings.Contains(str2, str1) {
                fmt.Println("The first string is a substring of the second 
string.")
        } else {
                fmt.Println("The first string is not a substring of the 
second string.")
        }
}
```

A2_A1

WAP in go language to print addition of two number using function.

```
package main

import "fmt"

func add(a, b int) int {
    return a + b
}

func main() {
    var num1, num2 int
    fmt.Print("Enter first number: ")
    fmt.Scanln(&num1)
    fmt.Print("Enter second number: ")
    fmt.Scanln(&num2)
    sum := add(num1, num2)
    fmt.Println("Sum:", sum)
}
```

A2_A2

WAP in go language to print recursive sum of digits of given number.

```
package main

import "fmt"

func sumOfDigits(n int) int {
    if n == 0 {
        return 0
    }
    return (n % 10) + sumOfDigits(n / 10)
}

func main() {
    num := 12345
    result := sumOfDigits(num)
    fmt.Println("The sum of digits of", num, "is", result)
}
```

A2_A3

WAP in go language using function to check whether accepts number is palindrome or not.

```
package main

import "fmt"

func isPalindrome(n int) bool {
    original := n
    reverse := 0
    for n > 0 {
        lastDigit := n % 10
        reverse = reverse*10 + lastDigit
        n = n / 10
    }
    return original == reverse
}

func main() {
    num := 121
    result := isPalindrome(num)
    if result {
        fmt.Println(num, "is a palindrome")
    } else {
        fmt.Println(num, "is not a palindrome")
    }
}
```

A2_B1

WAP in go language to swap two numbers using call by reference concept.

```
package main

import "fmt"

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

func main() {
    x := 5
    y := 3
    fmt.Println("Before swapping: x =", x, "y =", y)
    swap(&x, &y)
    fmt.Println("After swapping: x =", x, "y =", y)
}
```
A2_B2

WAP in go language to demonstrate use of names returns variables

```
package main

import "fmt"

func rectangleArea(length, width int) (area int) {
    area = length * width
    return
}

func main() {
    l := 10
    w := 5
    a := rectangleArea(l, w)
    fmt.Println("Area of rectangle with length", l, "and width", w, "is",
a)
}
```

A2_B3

WAP in go language to show the compiler throws an error if a variable is declared but not used.

```
package main

func main() {
    var x int
    // x is declared but not used
}
```

A2_C1

```
package main

import "fmt"

func change(num int) {
    num++
}

func main() {
    x := 5
    fmt.Println("Before calling change:", x)
    change(x)
    fmt.Println("After calling change:", x)
}
```

A2_C1

WAP in go language to illustrate the concept of call by value.

```
package main

import "fmt"

func changeValue(x int) {
    x = 10
    fmt.Println("Inside the function: x =", x)
}

func main() {
    num := 5
    fmt.Println("Before calling the function: num =", num)
    changeValue(num)
    fmt.Println("After calling the function: num =", num)
}
```

A2_C2

WAP in go language to create a file and write hello world in it and close the file by using defer statement.

```
package main

import (
        "fmt"
        "os"
)

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

        defer file.Close()

        _, err = file.WriteString("Hello, World!")
        if err != nil {
                fmt.Println("Error writing to file:", err)
                return
        }

        fmt.Println("Successfully wrote to file.")
}
```

A2_C3

WAP in go language to illustrate the concept of returning multiple values from a function

```go
package main

import "fmt"

func calculate(num1 int, num2 int) (int, int) {
    sum := num1 + num2
    diff := num1 - num2
    return sum, diff
}

func main() {
    x := 5
    y := 3
    sum, diff := calculate(x, y)
    fmt.Println("Sum:", sum)
    fmt.Println("Difference:", diff)
}
```

A3_A1

WAP in go language to find the largest and smallest number in an array

```go
import "fmt"

func findLargestAndSmallest(numbers []int) (int, int) {
	largest := numbers[0]
	smallest := numbers[0]

	for _, num := range numbers {
		if num > largest {
			largest = num
		}

		if num < smallest {
			smallest = num
		}
	}

	return largest, smallest
}

func main() {
	numbers := []int{3, 7, 2, 9, 10, 4}
	largest, smallest := findLargestAndSmallest(numbers)
	fmt.Println("Largest number:", largest)
	fmt.Println("Smallest number:", smallest)
}
```

A3_A2

go program to accept the book details such as bookid,title,author,price.
read and display the details of n number of books

```go
package main

import (
	"bufio"
	"fmt"
	"os"
	"strconv"
	"strings"
)

type Book struct {
	bookid int
	title  string
	author string
	price  float64
}

func readBookDetails(n int) []Book {
	var books []Book
	reader := bufio.NewReader(os.Stdin)

	for i := 0; i < n; i++ {
		fmt.Println("Enter details of book", i+1)
		fmt.Print("Enter bookid: ")
		bookid, _ := strconv.Atoi(readInput(reader))

		fmt.Print("Enter title: ")
		title := readInput(reader)

		fmt.Print("Enter author: ")
		author := readInput(reader)

		fmt.Print("Enter price: ")
		price, _ := strconv.ParseFloat(readInput(reader), 64)

		books = append(books, Book{bookid, title, author, price})
	}

	return books
}

func readInput(reader *bufio.Reader) string {
	input, _ := reader.ReadString('\n')
	input = strings.TrimSpace(input)
	return input
}

func displayBookDetails(books []Book) {
	fmt.Println("Book details:")
	for i, book := range books {
		fmt.Println("Book", i+1)
		fmt.Println("bookid:", book.bookid)
		fmt.Println("title:", book.title)
		fmt.Println("author:", book.author)
		fmt.Println("price:", book.price)
		fmt.Println()
	}
}

func main() {
	fmt.Print("Enter the number of books: ")
	reader := bufio.NewReader(os.Stdin)
	n, _ := strconv.Atoi(readInput(reader))

	books := readBookDetails(n)
	displayBookDetails(books)
}
```

A3_A3

Wap in go program to initialize a slice using multi-line syntax and display

```
package main

import "fmt"

func main() {
	numbers := []int{
		1, 2, 3,
		4, 5, 6,
		7, 8, 9,
	}

	fmt.Println("The slice is:", numbers)
}
```

A3_B1

go program to create and print multidimentional slice

```
package main

import "fmt"

func main() {
	matrix := [][]int{
		{1, 2, 3},
		{4, 5, 6},
		{7, 8, 9},
	}

	fmt.Println("The matrix is:")
	for _, row := range matrix {
		fmt.Println(row)
```

```
        }
}
```

A3_B2

go program to sort array elements in ascending order

```
package main

import (
        "fmt"
        "sort"
)

func main() {
        numbers := []int{5, 2, 9, 1, 7, 4, 8, 6, 3}

        fmt.Println("Unsorted numbers:", numbers)

        sort.Ints(numbers)

        fmt.Println("Sorted numbers:", numbers)
}
```

A3_B3

go program to accept n student details like rollno,student name.
mark1,mark2,mark3.calculate the total and average of marks using structure

```
package main

import "fmt"

type student struct {
        rollNo int
        name   string
        marks1, marks2, marks3 int
}

func (s student) total() int {
        return s.marks1 + s.marks2 + s.marks3
}

func (s student) average() float64 {
        return float64(s.total()) / 3
}

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

        students := make([]student, n)
```

```
	for i := 0; i < n; i++ {
		fmt.Printf("Enter details of student %d\n", i+1)
		fmt.Print("Roll No: ")
		fmt.Scan(&students[i].rollNo)
		fmt.Print("Name: ")
		fmt.Scan(&students[i].name)
		fmt.Print("Marks1: ")
		fmt.Scan(&students[i].marks1)
		fmt.Print("Marks2: ")
		fmt.Scan(&students[i].marks2)
		fmt.Print("Marks3: ")
		fmt.Scan(&students[i].marks3)
		fmt.Println()
	}

	for _, s := range students {
		fmt.Printf("Details of student %d\n", s.rollNo)
		fmt.Println("Name:", s.name)
		fmt.Println("Total:", s.total())
		fmt.Println("Average:", s.average())
		fmt.Println()
	}
}
```

A3_C1

go program to accept two matrices and display its multiplication

```
package main

import "fmt"

func main() {
	var rows1, cols1, rows2, cols2 int

	fmt.Print("Enter the number of rows for matrix 1: ")
	fmt.Scan(&rows1)
	fmt.Print("Enter the number of columns for matrix 1: ")
	fmt.Scan(&cols1)
	fmt.Print("Enter the number of rows for matrix 2: ")
	fmt.Scan(&rows2)
	fmt.Print("Enter the number of columns for matrix 2: ")
	fmt.Scan(&cols2)

	if cols1 != rows2 {
		fmt.Println("Matrix multiplication is not possible.")
		return
	}

	fmt.Println("Enter the elements of matrix 1:")
	matrix1 := make([][]int, rows1)
	for i := range matrix1 {
		matrix1[i] = make([]int, cols1)
		for j := range matrix1[i] {
			fmt.Printf("Enter the element [%d][%d]: ", i, j)
			fmt.Scan(&matrix1[i][j])
		}
	}

	fmt.Println("Enter the elements of matrix 2:")
	matrix2 := make([][]int, rows2)
	for i := range matrix2 {
		matrix2[i] = make([]int, cols2)
		for j := range matrix2[i] {
			fmt.Printf("Enter the element [%d][%d]: ", i, j)
			fmt.Scan(&matrix2[i][j])
		}
	}

	result := make([][]int, rows1)
	for i := range result {
		result[i] = make([]int, cols2)
	}

	for i := 0; i < rows1; i++ {
		for j := 0; j < cols2; j++ {
			sum := 0
			for k := 0; k < cols1; k++ {
				sum += matrix1[i][k] * matrix2[k][j]
			}
			result[i][j] = sum
		}
	}

	fmt.Println("Result:")
	for i := range result {
		for j := range result[i] {
			fmt.Print(result[i][j], " ")
		}
		fmt.Println()
	}
}
```

A3_C2

go program to accept n records of employee information (eno,ename,salary) and display record of emloyees having maximum 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)

	var employees []Employee
	for i := 0; i < n; i++ {
		var emp Employee
		fmt.Print("Enter employee no: ")
		fmt.Scan(&emp.eno)
		fmt.Print("Enter employee name: ")
		fmt.Scan(&emp.ename)
		fmt.Print("Enter employee salary: ")
		fmt.Scan(&emp.salary)
		employees = append(employees, emp)
	}

	maxSalary := 0
	var maxSalaryEmp Employee
	for _, emp := range employees {
		if emp.salary > maxSalary {
			maxSalary = emp.salary
			maxSalaryEmp = emp
		}
	}

	fmt.Println("Employee with maximum salary:")
	fmt.Println("Employee number:", maxSalaryEmp.eno)
	fmt.Println("Employee name:", maxSalaryEmp.ename)
	fmt.Println("Employee salary:", maxSalaryEmp.salary)
}
```

A3_C3

go program to demonstrate working of slices (like append,remove,copy etc.)

```
package main

import "fmt"

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

	// Append a value to the slice
	numbers = append(numbers, 6)
	fmt.Println("After appending 6:", numbers)

	// Remove an element from the slice
	numbers = append(numbers[:2], numbers[3:]...)
	fmt.Println("After removing the 3rd element:", numbers)

	// Copy a slice to another slice
```

```
        copyOfNumbers := make([]int, len(numbers))
        copy(copyOfNumbers, numbers)
        fmt.Println("Copy of slice:", copyOfNumbers)
}
```

A4_A1

go program to create an interface shape that includes area and perimeter. implements these methods in circle and rectangle type

```
package main

import (
        "fmt"
        "math"
)

// Shape is an interface that defines methods for calculating the area and
perimeter of a shape.
type Shape interface {
        Area() float64
        Perimeter() float64
}

// Circle represents a circle with a given radius.
type Circle struct {
        Radius float64
}

// Area returns the area of the circle.
func (c Circle) Area() float64 {
        return math.Pi * c.Radius * c.Radius
}

// Perimeter returns the circumference of the circle.
func (c Circle) Perimeter() float64 {
        return 2 * math.Pi * c.Radius
}

// Rectangle represents a rectangle with a given width and height.
type Rectangle struct {
        Width  float64
        Height float64
}

// Area returns the area of the rectangle.
func (r Rectangle) Area() float64 {
        return r.Width * r.Height
}

// Perimeter returns the perimeter of the rectangle.
func (r Rectangle) Perimeter() float64 {
        return 2 * (r.Width + r.Height)
}

func main() {
	c := Circle{Radius: 5}
	fmt.Println("Circle")
	fmt.Println("Radius: ", c.Radius)
	fmt.Println("Area: ", c.Area())
	fmt.Println("Perimeter: ", c.Perimeter())

	r := Rectangle{Width: 10, Height: 20}
	fmt.Println("\nRectangle")
	fmt.Println("Width: ", r.Width)
	fmt.Println("Height: ", r.Height)
	fmt.Println("Area: ", r.Area())
	fmt.Println("Perimeter: ", r.Perimeter())
}
```

A4_A2

go program to print multiplication of two numbers using method

```
package main

import "fmt"

// Number represents a number with a value.
type Number struct {
	Value int
}

// Multiply returns the multiplication of two numbers.
func (n Number) Multiply(m int) int {
	return n.Value * m
}

func main() {
	a := Number{Value: 5}
	b := 10
	result := a.Multiply(b)
	fmt.Println("Result:", result)
}
```

A4_A3

go program to create structure author . write a method show() whose receiver is struct author

```
package main

import "fmt"

// Author represents an author with a name and email.
type Author struct {
	Name  string
	Email string
}

// Show displays the details of an author.
func (a Author) Show() {
	fmt.Println("Name:", a.Name)
	fmt.Println("Email:", a.Email)
}

func main() {
	author := Author{Name: "John Doe", Email: "john.doe@example.com"}
	author.Show()
}
```

A4_B1

write a program in go language to create a structure student. write a method show() whose receiver is a pointer of struct student.

```
package main

import (
    "fmt"
)

type student struct {
    name   string
    rollNo int
    grade  string
}

func (s *student) show() {
    fmt.Printf("Name: %s\nRoll No: %d\nGrade: %s\n", s.name, s.rollNo,
s.grade)
}

func main() {

    s := student{
        name:   "John",
        rollNo: 1,
        grade:  "A",
    }

    s.show()
}
```

A4_B2

go program to demonstrate working type switch in interface.

```
package main

import (
    "fmt"
)

type geometry interface {
    area() float64
}

type rectangle struct {
    width, height float64
}

type circle struct {
    radius float64
}

func (r rectangle) area() float64 {
    return r.width * r.height
}

func (c circle) area() float64 {
    return 3.14 * c.radius * c.radius
}

func printArea(g geometry) {
    switch shape := g.(type) {
    case rectangle:
        fmt.Printf("Rectangle area: %f\n", shape.area())
    case circle:
        fmt.Printf("Circle area: %f\n", shape.area())
    default:
        fmt.Printf("Unknown shape\n")
    }
}

func main() {
    r := rectangle{width: 5, height: 10}
    c := circle{radius: 7}

    printArea(r)
    printArea(c)
}
```

A4_B3

go program to copy all elements of one array into another array using method.

```
package main

import "fmt"

func copyArray(src [5]int, dest [5]int) [5]int {
    for i := 0; i < len(src); i++ {
        dest[i] = src[i]
    }
    return dest
}

func main() {

    src := [5]int{1, 2, 3, 4, 5}
    dest := [5]int{}


    dest = copyArray(src, dest)

    fmt.Printf("Destination array: %v\n", dest)
}
```

A4_C1

go program to create an interface and display it's values with the help of type assertion.

```
package main

import "fmt"

type MyInterface interface {
    Display()
}

type MyStruct struct {
    message string
}

func (s MyStruct) Display() {
    fmt.Println(s.message)
}

func main() {
    mySlice := []MyInterface{
        MyStruct{message: "Hello, world!"},
        "This is a string",
       "This is another string",

    }


    for _, value := range mySlice {
        switch v := value.(type) {
        case MyStruct:
            s.Display()
        case string:
            fmt.Println(v)
        }
    }
}
```

A4_C2

go program to store n student information (rollno,name,percentage) and write a method to display student information in decending order of percentage.

```
package main

import (
    "fmt"
    "sort"
)

type Student struct {
    rollNo     int
    name       string
    percentage float64
}

type ByPercentage []Student

func (s ByPercentage) Len() int {
    return len(s)
}

func (s ByPercentage) Swap(i, j int) {
    s[i], s[j] = s[j], s[i]
}

func (s ByPercentage) Less(i, j int) bool {
    return s[i].percentage > s[j].percentage
}

func DisplayStudents(students []Student) {
    sort.Sort(ByPercentage(students))
    for _, student := range students {
        fmt.Printf("Roll No: %d, Name: %s, Percentage: %.2f%%\n",
student.rollNo, student.name, student.percentage)
    }
}

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

    students := make([]Student, n)

    for i := 0; i < n; i++ {
        fmt.Printf("Enter the information for student %d:\n", i+1)
        fmt.Print("Roll No: ")
        fmt.Scanln(&students[i].rollNo)
        fmt.Print("Name: ")
        fmt.Scanln(&students[i].name)
        fmt.Print("Percentage: ")
        fmt.Scanln(&students[i].percentage)
    }

    DisplayStudents(students)
}
```

A4_C3

go program to demonstrate working embedded interfaces.

```
package main

import (
    "fmt"
)

type Animal interface {
    Speak() string
}

type Pet interface {
    Name() string
    SetName(name string)
}

type Dog struct {
    name string
}

func (d Dog) Speak() string {
    return "Woof!"
}

func (d *Dog) Name() string {
    return d.name
}

func (d *Dog) SetName(name string) {
    d.name = name
}

func NameAndSpeak(pet Pet, animal Animal) {
    pet.SetName("Fido")
    fmt.Printf("The %T named %s says %q\n", animal, pet.Name(),
animal.Speak())
}

func main() {
    d := new(Dog)
    NameAndSpeak(d, d)
}
```

A5_A1

go program using go routine and channel that will printthe sum of the squares and cubes of the individual digits of a number.

```go
package main

import (
    "fmt"
    "strconv"
)

func main() {
    number := 12345
    digits := strconv.Itoa(number)
    sq := make(chan int)
    cu := make(chan int)
    go calcSquares(digits, sq)
    go calcCubes(digits, cu)
    sumSquares, sumCubes := <-sq, <-cu
    fmt.Println("Sum of squares:", sumSquares)
    fmt.Println("Sum of cubes:", sumCubes)
}

func calcSquares(digits string, sq chan int) {
    sum := 0
    for _, digit := range digits {
        num, _ := strconv.Atoi(string(digit))
        sum += num * num
    }
    sq <- sum
}

func calcCubes(digits string, cu chan int) {
    sum := 0
    for _, digit := range digits {
        num, _ := strconv.Atoi(string(digit))
        sum += num * num * num
    }
    cu <- sum
}
```

A5_A2

go program that executes 5 go routines simultaneously which generates numbers from 0 to 10 waiting between 0 to 250 ms after each go routine.

```go
package main

import (
    "fmt"
    "math/rand"
    "time"
)
```

```
func main() {
    for i := 1; i <= 5; i++ {
        go generate(i)
    }
    time.Sleep(time.Second)
}

func generate(id int) {
    rand.Seed(time.Now().UnixNano())
    for i := 0; i <= 10; i++ {
        fmt.Printf("Goroutine %d: %d\n", id, i)
        time.Sleep(time.Duration(rand.Intn(250)) * time.Millisecond)
    }
}
```

A5_A3

write a go program that creates a slice of integers, checks numbers from slice are even or odd and further sent to respective go routines through channel and display values received by go routines.

```
package main

import (
    "fmt"
)

func main() {
    nums := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
    evenChan := make(chan int)
    oddChan := make(chan int)

    for _, num := range nums {
        if num%2 == 0 {
            go sendEven(num, evenChan)
        } else {
            go sendOdd(num, oddChan)
        }
    }

    for i := 0; i < len(nums)/2; i++ {
        fmt.Printf("Even Goroutine: %d\n", <-evenChan)
        fmt.Printf("Odd Goroutine: %d\n", <-oddChan)
    }
}

func sendEven(num int, c chan<- int) {
    c <- num
}

func sendOdd(num int, c chan<- int) {
    c <- num
}
```

A5_B1

go program to create buffered channel,store few values in it and find channel capacity and length. read values from channel and find modified length of a channel.

```
package main

import (
    "fmt"
)

func main() {
    c := make(chan int, 3)
    c <- 1
    c <- 2
    c <- 3
    fmt.Printf("Channel capacity: %d\n", cap(c))
    fmt.Printf("Channel length: %d\n", len(c))
    x := <-c
    y := <-c
    fmt.Printf("Channel length after reading 2 values: %d\n", len(c))
    z := <-c
    close(c)
    fmt.Printf("Channel length after reading all values and closing 
channel: %d\n", len(c))
    fmt.Printf("Values received from channel: %d, %d, %d\n", x, y, z)
}
```

A5_B2

write a program in go main go routine to to read and write fibonacci series to the channel.

```
package main

import "fmt"

func fibonacci(c chan int, n int) {
    x, y := 0, 1
    for i := 0; i < n; i++ {
        c <- x
        x, y = y, x+y
    }
    close(c)
}

func main() {
    c := make(chan int)
    go fibonacci(c, 10)
    for x := range c {
        fmt.Println(x)
    }
}
```

A5_B3

go program for how to create channel and illustrate how to close a channel using for range loop and close function.

```
package main

import "fmt"

func main() {
    c := make(chan string) // create a channel of type string
    go func() {
        c <- "hello" // send a value to the channel
        c <- "world" // send another value to the channel
        close(c)     // close the channel
    }()
    for s := range c {
        fmt.Println(s) // print each value received from the channel
    }
}
```

A5_C1

Write a go program to implement the checkpoint synchronization problem which is a
problem of synchronizing multiple tasks. Consider a workshop where several workers assembling details of some mechanism. When each of them completes his work,
they put the details together. There is no store, so a worker who finished its part first must wait for others
before starting another one. Putting details together is the checkpoint at which tasks
synchronize themselves before going their paths apart.
*/
//--

```
package main

import (
        "log"
        "math/rand"
        "sync"
        "time"
)

func worker(part string) {
        log.Println(part, "worker begins part")
        time.Sleep(time.Duration(rand.Int63n(1e6)))
        log.Println(part, "worker completes part")
        wg.Done()
}

var (
        partList    = []string{"A", "B", "C", "D"}
        nAssemblies = 3
        wg          sync.WaitGroup
)

func main() {
        rand.Seed(time.Now().UnixNano())
        for c := 1; c <= nAssemblies; c++ {
                log.Println("begin assembly cycle", c)
                wg.Add(len(partList))
                for _, part := range partList {
                        go worker(part)
                }
                wg.Wait()
                log.Println("assemble.  cycle", c, "complete")
        }
}
```

A6_A1

write a go program to create student struct with student name and marks and sort it based on student marks using sort package

```
package main

import (
        "fmt"
        "sort"
)

type student struct {
        name  string
        marks int
}

func main() {
        // Create a slice of student structs
        students := []student{
                {name: "Amol", marks: 85},
                {name: "amey", marks: 92},
                {name: "shubh", marks: 78},
                {name: "rohan", marks: 91},
                {name: "harshal", marks: 87},
        }

        // Sort the slice based on student marks
        sort.Slice(students, func(i, j int) bool {
                return students[i].marks > students[j].marks
        })

        // Print the sorted slice
        fmt.Println("Sorted students:")
        for _, s := range students {
                fmt.Printf("%s: %d\n", s.name, s.marks)
        }
}
```

A6_A2

go program using user defined package calculator that performs one calculator operation as per the user's choice

```
package main

import (
	"fmt"
	"os"
	"strconv"

	"calculator" // Import the user-defined calculator package
)

func main() {
	if len(os.Args) != 4 {
		fmt.Println("Usage: calculator <number> <operation>
<number>")
		return
	}

	num1, err := strconv.ParseFloat(os.Args[1], 64)
	if err != nil {
		fmt.Println("Invalid first number:", os.Args[1])
		return
	}

	num2, err := strconv.ParseFloat(os.Args[3], 64)
	if err != nil {
		fmt.Println("Invalid second number:", os.Args[3])
		return
	}

	var result float64
	switch os.Args[2] {
	case "+":
		result = calculator.Add(num1, num2)
	case "-":
		result = calculator.Subtract(num1, num2)
	case "*":
		result = calculator.Multiply(num1, num2)
	case "/":
		result = calculator.Divide(num1, num2)
	default:
		fmt.Println("Invalid operation:", os.Args[2])
		return
	}

	fmt.Printf("%.2f %s %.2f = %.2f\n", num1, os.Args[2], num2, result)
}
```

A6_A3

WAP in Go language to create an user defined package to find out the area of a rectangle.

```
package main

import (
	"fmt"

	"rectangle" // Import the user-defined rectangle package
)

func main() {
	r := rectangle.Rectangle{Length: 4, Width: 6}

	area := rectangle.Area(r)

	fmt.Printf("The area of the rectangle is %.2f square units.\n",
area)
}
```

A6_B1

go program to add two integers and write code for unit test to test this code

```
package main

import (
	"fmt"
	"testing"
)

func Add(x, y int) int {
	return x + y
}

func main() {
	sum := Add(3, 5)
	fmt.Println("The sum of 3 and 5 is", sum)
}

// Unit test for the Add function
func TestAdd(t *testing.T) {
	tests := []struct {
		x, y int
		want int
	}{
		{2, 3, 5},
		{-1, 1, 0},
		{0, 0, 0},
		{2147483647, 1, -2147483648}, // Test integer overflow
	}
	for _, test := range tests {
		got := Add(test.x, test.y)
		if got != test.want {
			t.Errorf("Add(%d, %d) = %d; want %d", test.x,
test.y, got, test.want)
		}
	}
}
```

A6_B2

program to substract two integers and write code for table test to test this code

```
package main

import (
	"fmt"
	"testing"
)

func Subtract(x, y int) int {
	return x - y
}

func main() {
	diff := Subtract(5, 3)
	fmt.Println("The difference between 5 and 3 is", diff)
}

// Table test for the Subtract function
func TestSubtract(t *testing.T) {
	tests := []struct {
		x, y int
		want int
	}{
		{5, 3, 2},
		{3, 5, -2},
		{0, 0, 0},
		{-5, -3, -2},
		{2147483647, -1, -2147483648}, // Test integer overflow
	}
	for _, test := range tests {
		got := Subtract(test.x, test.y)
		if got != test.want {
			t.Errorf("Subtract(%d, %d) = %d; want %d", test.x,
test.y, got, test.want)
		}
	}
}
```

A6_B3

write a function in go language to find the square of a number and write a benchmark for it

```
package main

import (
	"fmt"
	"testing"
)

func Square(x int) int {
	return x * x
}

func main() {
	x := 5
	sq := Square(x)
	fmt.Printf("The square of %d is %d\n", x, sq)
}

// Benchmark for the Square function
func BenchmarkSquare(b *testing.B) {
	for i := 0; i < b.N; i++ {
		Square(5)
	}
}
```

A6_C1

program to read a xml file into structure and display structure

```
package main

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

type Person struct {
	XMLName xml.Name `xml:"person"`
	Name    string   `xml:"name"`
	Age     int      `xml:"age"`
	City    string   `xml:"city"`
}

func main() {
	xmlFile, err := os.Open("person.xml")
	if err != nil {
		fmt.Println("Error opening file:", err)
		return
	}
	defer xmlFile.Close()
	byteValue, err := ioutil.ReadAll(xmlFile)
	if err != nil {
		fmt.Println("Error reading file:", err)
		return
	}

	var person Person
	err = xml.Unmarshal(byteValue, &person)
	if err != nil {
		fmt.Println("Error unmarshalling XML:", err)
		return
	}

	fmt.Printf("Name: %s\nAge: %d\nCity: %s\n", person.Name,
person.Age, person.City)
}
```

A6_C2

program to print file information

```
package main

import (
	"fmt"
	"os"
)

func main() {
	// Open the file
	file, err := os.Open("filename.txt")
	if err != nil {
		fmt.Println("Error:", err)
		return
	}
	defer file.Close()

	// Get file information
	fileInfo, err := file.Stat()
	if err != nil {
		fmt.Println("Error:", err)
		return
	}

	// Print file information
	fmt.Println("Name:", fileInfo.Name())
	fmt.Println("Size:", fileInfo.Size(), "bytes")
	fmt.Println("Mode:", fileInfo.Mode())
	fmt.Println("Modified:", fileInfo.ModTime())
}
```

A6_C3

go program to add or append content at the end of a text file

```
package main

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

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

	// Ask the user for input to append
	reader := bufio.NewReader(os.Stdin)
	fmt.Print("Enter text to append: ")
	text, err := reader.ReadString('\n')
	if err != nil {
		fmt.Println("Error:", err)
		return
	}

	// Append the text to the file
	_, err = file.WriteString(text)
	if err != nil {
		fmt.Println("Error:", err)
		return
	}

	fmt.Println("Text appended successfully!")
}
```

```go
package main

import "fmt"

func calculate(num1 int, num2 int) (int, int) {
    sum := num1 + num2
    diff := num1 - num2
    return sum, diff
}

func main() {
    x := 5
    y := 3
    sum, diff := calculate(x, y)
    fmt.Println("Sum:", sum)
    fmt.Println("Difference:", diff)
}
```