GO

20m

1.

A) WAP to accept user choice and print answers using arithmetic operators.

```
package main
import "fmt"
func main(){
    var a,b,n int
    fmt.Print("Enter a value of a :")
    fmt.Scanf("%d\n",&a)
    fmt.Print("Enter a value of b :")
    fmt.Scanf("%d\n",&b)
    fmt.Printf("1.Addtion\n2.Substriction\n3.Multiplication\n4.Divison\nEnter a case:")
    fmt.Scanf("%d",&n)

    switch (n){
    case 1:
            fmt.Printf("%d + %d = %d\n",a,b,a+b)
    case 2:
            fmt.Printf("%d - %d = %d\n",a,b,a-b)
    case 3:
            fmt.Printf("%d * %d = %d\n",a,b,a*b)
    case 4:
            fmt.Printf("%d + %d = %d\n",a,b,a/b)
    default :
            fmt.Println("Invalid case")
    }
}
```

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 isPalindrome(n int) int {
        rev := 0
        for n > 0 {
            rem := n% 10
            rev = rev*10 + rem
            n= n / 10
        }
        return    rev
  }

func main() {
    var num , rev int
    fmt.Print("Enter a number:")
    fmt.Scanf("%d",&num)
    rev = isPalindrome(num)
    if(num==rev) {
            fmt.Printf("%d is a palindrome number",num)
    }else {
            fmt.Printf("%d is not a palindrome number",num)
    }
}
```

4.
A) WAP to print a recursive sum of digits of a given number.

```
package main
import "fmt"

func recurSum(n int) int{
      if n<10 {
            return n
      }
      return n%10 + recurSum(n/10)
  }

func main() {
      var num int
      fmt.Print("Enter a number:")
      fmt.Scanf("%d",&num)
      sum := recurSum(num)
      fmt.Printf("Recursive sum of digits of %d is %d\n",num,sum)
}
```

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 float64
}

func main() {
              var n int
              var emp [10]Employee

              fmt.Print("How many employee details you wants to enter : ")
              fmt.Scan(&n)
```

```
        fmt.Println("Enter Details : ")

        for i := 0; i < n; i++ {
                fmt.Println("Employee ", i+1)
                fmt.Print("Employee No. : ")
                fmt.Scan(&emp[i].eno)
                fmt.Print("Employee Name : ")
                fmt.Scan(&emp[i].ename)
                fmt.Print("Employee Salary : ")
                fmt.Scan(&emp[i].salary)
        }

        loc := 0
        minSalary := emp[0].salary

        for i := 1; i < n; i++ {
                if minSalary > emp[i].salary {
                        minSalary = emp[i].salary
                        loc = i
                }
        }

        fmt.Println("Employee having minimum salary : \n")
        fmt.Println("Employee No. : ", emp[loc].eno)
        fmt.Println("Employee Name : ", emp[loc].ename)
        fmt.Println("Employee Salary : ", emp[loc].salary)
}
```

6.
B) WAP to copy all elements of one array into another using a method.

```
package main
import "fmt"

type Array []int

func (a Array) copyTo (b Array){
     for i,v := range a{
        b[i] = v
     }
}

func main(){
       var n int
       fmt.Println("Enter size")
       fmt.Scan(&n)
           a:=make(Array ,n)
           fmt.Println("Enter elements")
           for i:=0;i<n;i++{
                fmt.Scan(&a[i])
           }
           b:=make(Array ,len(a))
           a.copyTo(b)
           fmt.Println("Array a :",a)
           fmt.Println("Array b :",b)
}
```

7.
B) WAP to create structure student. Write a method show() whose receiver is a pointer of struct student.

```
package main
import "fmt"

type student struct {
        roll    int
        name    string
        marks float64
}

func (s *student) show(){
        fmt.Println("\nRoll No. : ", s.roll)
        fmt.Println("Name : ", s.name)
        fmt.Println("Marks : ", s.marks)
}
func main() {
        var s [10]student
        var n int
        fmt.Print("Enter No. of students ")
        fmt.Scan(&n)
        for i:=0;i<n;i++{
            fmt.Printf("Enter Details of Student %d\n",i+1)
            fmt.Println("Enter Roll no:")
            fmt.Scan(&s[i].roll)
            fmt.Println("Enter Name:")
            fmt.Scan(&s[i].name)
            fmt.Println("Enter Marks:")
            fmt.Scan(&s[i].marks)
         }

        for i:=0;i<n;i++{
            fmt.Printf("Details of student %d is\n",i+1)
            s[i].show()
        }

}
```

8.
A) WAP to accept the book details such as BookID, Title, Author, Price. Read and display the details of 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 to input: ")
	fmt.Scan(&n)

	// Create an array of book structs with size n
	books := make([]book, n)

	for i := 0; i < n; i++ {
		fmt.Printf("Enter details for 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()
	}

	fmt.Println("Details for each book:")
	for i := 0; i < n; i++ {
		fmt.Printf("Book ID: %d\n", books[i].bookID)
		fmt.Printf("Title: %s\n", books[i].title)
		fmt.Printf("Author: %s\n", books[i].title)
		fmt.Printf("Price: %.2f\n",books[i].price)
		fmt.Println()
	}
}
```

9.
A) WAP using a function to check whether the accepted number is palindrome or not.
 Refer slip 3

10.
A) WAP to create an interface and display its values with the help of type assertion.

```
package main
import "fmt"

func main() {
		var i interface{} = 5.2

		if v, result := i.(string); result {
				fmt.Println("Value is : ", v, "\nIt is a String ")
```

```
        } else if v, result := i.(int); result {
                fmt.Println("Value is : ", v, "\nIt is a Integer ")
        } else {
                v := i.(float64)
                fmt.Println("Value is : ", v, "\nIt is a Float")
        }

}
```

11.
A) WAP to check whether the accepted number is two digit or not.

```
package main
import "fmt"

func main(){
  var a int
  fmt.Print("Enter a number to check :")
  fmt.Scanf("%d",&a)

  if(a>=10 && a<=99){
        fmt.Print("The given number is two digit \n")
  }else{
        fmt.Print("The given number is not two digit\n")
  }
}
```

12.
A) WAP to swap two numbers using call by reference concept

```
package main
import "fmt"

func swap(x *int , y *int) {
  temp := *x
  *x = *y
  *y = temp
}

func main() {
  fmt.Println("Call by reference")
  var a,b int
  fmt.Println("Enter two numbers:")
  fmt.Scan(&a,&b)
  fmt.Printf("Before swapping: a = %d,b = %d\n",a,b)
  swap(&a,&b)
  fmt.Printf("After swapping:a = %d,b = %d\n",a,b)
}
```

14.
A) WAP to demonstrate working of slices (like append, remove, copy etc.)

```
package main
```

```
import   "fmt"

func main() {
            arr := []int{1, 2, 3, 4, 5, 6, 7, 8, 9}
            arr2 := make([]int, len(arr))

            fmt.Println("Slice : ", arr)

            arr = append(arr, 10)
            fmt.Println("\nAfter appending 10 in slice : ", arr)

            arr = arr[:len(arr)-1]
            fmt.Println("\nAfter removing last element in slice : ", arr)

            copy(arr2, arr)
            fmt.Println("\nCopying one slice into another one :", arr2)
}
```

16.
A) WAP to create a user defined package to find out the area of a rectangle.

rectangle.go package file

```
package rectangle

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

Main file

```
package main
import (
       "fmt"
       "rectangle"
)

func main() {
       length := 5.0
       width := 3.0
       area := rectangle.Area(length, width)
       fmt.Printf("The area of the rectangle with length %f and width %f is %f", length, width, area)
}
```

20.
A) WAP to add or append content at the end of a text file.

```
package main
import (
       "fmt"
       "os"
)
```

```
func main() {
	file, err := os.OpenFile("demo.txt", os.O_APPEND|os.O_WRONLY, 0644)
	if err != nil {
		fmt.Println(err)
		return
	}
	defer file.Close()

	_, err2 := file.WriteString("Hii\n")
	if err2 != nil {
		fmt.Println(err2)
		return
	}

	fmt.Println("Operation successful!")
}
```

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.

```
package main
import   "fmt"

func main() {
		// create a channel of integers
		ch := make(chan int)

		// start a goroutine to send some values to the channel
		go func() {
				for i := 1; i <= 5; i++ {
						ch <- i
				}
				// close the channel after sending all values
				close(ch)
		}()

		// use a for range loop to receive values from the channel
		for val := range ch {
				fmt.Println("Received value:", val)
		}

		// check if the channel is closed or not
		_, ok := <-ch
		if ok {
				fmt.Println("Channel is not closed")
		} else {
				fmt.Println("Channel is closed")
		}
}
```