Assignment 1

1. WAP in go language to print Student name, rollno, division and college name

```
package main
import "fmt"

func main() {
	fmt.Print("Enter Student Name : ")
	var name string
	fmt.Scanln(&name)

	fmt.Print("Enter Student Rollno : ")
	var roll int
	fmt.Scanln(&roll)

	fmt.Print("Enter Student Division : ")
	var Division string
	fmt.Scanln(&Division)

	fmt.Print("Enter Student College : ")
	var college string
	fmt.Scanln(&college)

	fmt.Println("Name : ", name)
	fmt.Println("roll : ", roll)
	fmt.Println("Division : ", Division)
	fmt.Println("College : ", college)
}
```

2. WAP in go language to print whether number is even or odd.

```
package main
import "fmt"

func main(){
   fmt.Print("Enter number : ")
   var n int
   fmt.Scanln(&n)
   /*  Conditional Statement if .... else .....     */
   if(n%2==0){
      fmt.Println(n,"is Even number")
   }else{
      fmt.Println(n,"is Odd number")
   }
}
```

3. WAP in go language to swap the number without temporary variable.

```
package main
import "fmt"

func main(){
  a := 23
  b := 45

  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)
}
```

4. WAP in go Language to print address of a variable.

```
package main
import "fmt"

func main() {

        // variable
        i := 32
        fmt.Println(&i)

        // pointer to the variable
        p := &i
        fmt.Println(p)
}
```

SETB

1. WAP in go to print table of given number.

```
package main
import "fmt"

func main(){
   var n int
   fmt.Print("Enter any Integer Number : ")
   fmt.Scan(&n)
   i:=1
   /*     For loop as a Go's While     */
   for {
      if(i>10){
         break;
      }
      fmt.Println(n," X ",i," = ",n*i)
      i++
   }
```

```
}
```

2. WAP in go language to print PASCALS triangle.

```
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("")
	}
}
```

3. 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)
    }
}
```

4. WAP in go language to illustrate pointer to pointer concept

```
package main
import "fmt"

func main() {
        var a int = 10
        var b *int
        var c **int
        b = &a
        c = &b
        fmt.Println("valuee of a = ", a)
        fmt.Println("valuee of b = ", *b)
        fmt.Println("valuee of c = ", **c)
}
```

5. WAP in go language to explain new function

```
package main
```

```
import "fmt"

func area(length, width int)int{

   Ar := length* width
   return Ar
}

func main() {
        var x,y int
        fmt.Print("Enter the length : ")
        fmt.Scan(&x)

        fmt.Print("Enter the Width : ")
        fmt.Scan(&y)

  fmt.Printf("Area of rectangle is : %d", area(x, y))
}
```

SETC

1. WAP in go language to concatenate two strings using pointers.

```
package main
import "fmt"

func main() {
        var concatenate string
        var a, b string
        fmt.Print("Enter the First String : ")
        fmt.Scan(&a)
        fmt.Print("Enter the second String : ")
        fmt.Scan(&b)

        // var b string =
        var c *string
        var d *string
```

```
        c = &a
        d = &b
        concatenate = *c + *d
        fmt.Println("Concatenated String : ", concatenate)
}
```

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

```
package main
import "fmt"

func main() {
    fmt.Print("Enter First String: ")
      //Print function is used to display output in same line
    var first string
    fmt.Scanln(&first)                    // Take input from user
      fmt.Print("Enter Second String: ")
      var second string
      fmt.Scanln(&second)

    if first == second {
        fmt.Println("Strings are equal !")
    }else{
        fmt.Println("Strings are not equal !")
    }
}
```

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

```
package main
import "fmt"

func main() {
        var x, y int
        fmt.Print(" Enter The Two Number  : ")
        fmt.Scan(&x, &y)
```

```
	var choice string
	fmt.Print("which operation you should perform : ")
	fmt.Scan(&choice)

	switch choice {
	case "+":
		fmt.Println("Addition of two number", x+y)
	case "-":
		fmt.Println("Subtraction of two number", x-y)
	case "*":
		fmt.Println("Multiplition of two number", x*y)
	case "/":
		fmt.Println("Divition of two number", x/y)
	case "%":
		fmt.Println("Modulation of two number", x%y)
	default:
		fmt.Println(" Enter the valid oprator ")
	}
}
```

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

```
package main
import "fmt"

func main() {
	var n int
	fmt.Println("Enter a number = ")
	fmt.Scanln(&n)

	if n >= 0 && n <= 9 {
		fmt.Println(" Single digit number = ", n)
	} else if n >= 10 && n <= 99 {
		fmt.Println("Double digit number =", n)
	} else if n >= 100 && n <= 999 {
```

```
            fmt.Println("Triple digit number = ", n)
        } else if n >= 1000 && n <= 9999 {
            fmt.Println("Four digit number = ", n)
        } else {
            fmt.Println("Five digit number = ", n)
        }

}
```

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

```
package main

import ("fmt"
    "strings" )


func main() {
    var string1, substring string = "Golang", "Go"


    if strings.Contains(string1, substring) {
        fmt.Println("first string is substring of another string")
    } else {
        fmt.Println("first string is not substring of another
string")


    }
}
```

Assignment 2

SETA

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

```
package main

import "fmt"


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


func main()  {
   var sum int
   var a, b int
        fmt.Print("Enter the First Number : ")
        fmt.Scan(&a)
        fmt.Print("Enter the Second Number : ")
        fmt.Scan(&b)


   sum = add(a,b)
   fmt.Println("The Sum of Two Numbers  = ", sum)
}
```

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

```
package main
import (
	"fmt"
)

func sumDigits(num int) int {
	if num == 0 {
		return 0
	} else {
		return ((num % 10) + sumDigits(num/10))
	}
}
func main() {
	var num int
	fmt.Print("Enter Number:")
	fmt.Scanln(&num)
	fmt.Printf("Addition digits of %d = %d\n", num,
sumDigits(num))

}
```

3. WAP in 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")
    }
}
```

SETB

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

```
package main
import "fmt"

func main() {
    var a,b int
    fmt.Print("Enter First Number : ")
    fmt.Scanln(&a)

    fmt.Print("Enter Second Number : ")
    fmt.Scanln(&b)

    fmt.Println("Before Swap value a is ", a)
    fmt.Println("Before Swap value b is ", b)
    swap(&a ,&b)
    fmt.Println("After Swap value a is ", a)
    fmt.Println("After Swap value b is ", b)
}
```

```
func swap(x *int, y *int){
   var temp int
      temp = *x /* save the value at address x */
      *x = *y /* put y into x */
      *y = temp /* put temp into y */
  }
```

2. WAP in go language to demonstrate use of names returns variables.

```
package main
import "fmt"

func split(sum int) (x, y int) {
        x = sum * 4 / 9
        y = sum - x
        return
}

func main() {
        fmt.Println(split(18))
}
```

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

```
package main
import "fmt"

func main() {

        i := 1
}
```

SETC

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

```
package main
import "fmt"

func main() {
	var a, b int
	fmt.Print("Enter First Number : ")
	fmt.Scanln(&a)

	fmt.Print("Enter Second Number : ")
	fmt.Scanln(&b)

	fmt.Println("Before Swap value a is ", a)
	fmt.Println("Before Swap value b is ", b)
	swap(a, b)
	fmt.Println("After Swap value a is ", a)
	fmt.Println("After Swap value b is ", b)

}

func swap(x int, y int) (int ,int) {
	var temp int
	temp = x /* save the value at address x */
	x = y    /* put y into x */
	y = temp /* put temp into y */
	return x,y
}
```

2. 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"
```

```
import "os"

func main() {
   f, _ := os.Create("hello.txt")
   defer f.Close()
   fmt.Fprintln(f,"hello world")
}
```

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

```
package main
import "fmt"

func myfunc(a, b int)(int, int, int ){
        return a - b, a * b, a + b
}

func main() {

var v1,v2,v3 int
fmt.Println("Enter 3 Values ")
fmt.Scan(&v1,&v2,&v3)

v1,v2,v3 = myfunc(v1, v2)

fmt.Printf("Result is: %d", v1 )
fmt.Printf("\nResult is: %d", v2)
fmt.Printf("\nResult is: %d", v3)
}
```

Assignment 3

SETA

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

```
package main
import "fmt"

func main() {
   var s, i int

   fmt.Print("Enter the Array Size to find Smallest & Largest = ")
   fmt.Scan(&s)

   lgsmArr := make([]int, s)

   fmt.Print("Enter the Array Items  = ")
   for i = 0; i < s; i++ {
      fmt.Scan(&lgsmArr[i])
   }
   largest := lgsmArr[0]
   smallest := lgsmArr[0]
```

```
   for i = 0; i < s; i++ {
      if largest < lgsmArr[i] {
         largest = lgsmArr[i]
      }
      if smallest > lgsmArr[i] {
         smallest = lgsmArr[i]
      }
   }
   fmt.Println("\nThe Largest Number in this Array    = ", largest)
   fmt.Println("\nThe Smallest Number in this lgsmArr    = ",
smallest)
}
```

2. WAP 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 bookdetails struct {
        id     int
        price  float64
        title,author  string
}
```

```
func main() {
	var n, i int
	var book[100] bookdetails

	fmt.Print("How many book details Enter : ")
	fmt.Scan(&n)

	for i = 0; i < n; i++ {
		fmt.Println("Enter the Book ID : ")
		fmt.Scan(&book[i].id)

		fmt.Println("Enter the Book Title : ")
		fmt.Scan(&book[i].title)

		fmt.Println("Enter the Book Author : ")
		fmt.Scan(&book[i].author)

		fmt.Println("Enter the Book Price : ")
		fmt.Scan(&book[i].price)
		fmt.Println(" ------------------------------ ******* ------------
--------------")

	}

	for i = 0; i < n; i++ {

		fmt.Println("Book ID : ", book[i].id)
		fmt.Println("Book Title : ", book[i].title)
		fmt.Println("Book Author : ", book[i].author)
		fmt.Println("Book Price : ", book[i].price)
		fmt.Println(" ------------------------------ ******* ------------
--------------")
	}

}
```

3. WAP in go language to Initialize a Slice using Multi-Line Syntax and display

```
package main
import "fmt"
func main(){
	a := [4] int {
		9,
		65,
		82,
		0,
	}
	fmt.Println(a)
}
```

SETB

1. WAP in go language to create and print multidimensional Slice

```
Package main
import "fmt"
func main(){
	s1 := [][]int {
		{1,2},
		{3,4},
		{5,6},
		{7,8},
	}
	fmt.Println("Slice 1 :" , s1)
}
```

2. WAP in go language to sort array elements in ascending order

```
package main
import "fmt"
func main(){
	var temp , s , i, j int
	fmt.Println("Enter the size of Arrary : ")
	fmt.Scanln(&s)
	a := make([]int, s)
	fmt.Println("Enter the Array Items : ")
	for i := 0; i < s; i++ {
		fmt.Scanln(&a[i])
	}
	for i = 0; i < s; i++ {
		for j = i+1; j < s; j++ {
			if a[i] > a[j] {
				temp = a[i]
				a[i] = a[j]
				a[j] = temp
			}
		}
	}
	fmt.Println("Arrary after ascending order :")
	for j = 0; j < s; j++ {
		fmt.Println(a[j])
	}

}
```

3. WAP 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 {
	srno	int
	m1, m2, m3 int
	sname	string
}

func main() {
	var s1 Student
	var total, avg float64

	fmt.Print("Enter the Student Roll No : ")
	fmt.Scan(&s1.srno)

	fmt.Println("Enter the Student Name : ")
	fmt.Scan(&s1.sname)

	fmt.Println("Enter the Student marks for 3 subjects :
")
	fmt.Scan(&s1.m1, &s1.m2, &s1.m3)

	fmt.Println(" ----------------------------- ******* ----------
-------------")

	fmt.Println("Roll No : ", s1.srno)

	fmt.Println("Name  : ", s1.sname)
	fmt.Println("Marks for 3 Subject  : ", s1.m1, s1.m2,
s1.m3)

	total = float64(s1.m1 + s1.m2 + s1.m3)

	avg = float64(total / 3)

	fmt.Println("Total Number : ", total)
```

```
        fmt.Println("Average Marks : ", avg)

        fmt.Println(" ---------------------------- ******* ----------
--------------")
}
```

SETC

1. WAP in go language to accept two matrices and display it's multiplication.

```
package main

import "fmt"

func main() {
        var m, n, p, q, total int
        var a [5][5]int
        var b [5][5]int
        var c [5][5]int

        fmt.Println("Enter the order of First Matrix :")
        fmt.Scanln(&m, &n)

        fmt.Println("Enter the order of Second matrix :")
        fmt.Scanln(&p, &q)

        if n != p {
                fmt.Println("Error : The matrix cannot be
multiplied")
        }else{
                fmt.Println("Enter The First Matrix ")
                for i := 0; i < m; i++ {
                        for j := 0; j < n; j++ {
                                fmt.Println("Enter Element :")
```

```
                        fmt.Scan(&a[i][j])
                }
        }

        fmt.Println("Enter The Second Matrix ")
        for i := 0; i < p; i++ {
                for j := 0; j < q; j++ {
                        fmt.Println("Enter Element :")
                        fmt.Scan(&b[i][j])
                }
        }

        for i := 0; i < m; i++ {
                for j := 0; j < p; j++ {
                        for k := 0; k < p; k++ {
                                total = total + a[i][k]* b[k][j]
                        }
                        c[i][j] = total
                        total = 0
                }
        }

        fmt.Println("Multiplication Matrix : ")
        for i := 0; i < m; i++ {
                for j := 0; j < n; j++ {
                        fmt.Print(" ",c[i][j])
                }
                fmt.Print("\n")
        }

    }
}
```

2. WAP in go language to accept n records of employee information (eno,ename,salary) and display record of employees having maximum salary.

```
package main
import "fmt"
type employee struct{
	eno int
	esal float64
	ename string
}
func main(){
	var e1[100] employee
	var n,k int
	var max float64
	fmt.Println("Enter No Of Employess you Want : ")
	fmt.Scan(&n)
	for i := 0; i < n; i++ {

		fmt.Println("Enter Employee No :")
		fmt.Scan(&e1[i].eno)

		fmt.Println("Enter Employee Name :")
		fmt.Scan(&e1[i].ename)

		fmt.Println("Enter Employee Salary :")
		fmt.Scan(&e1[i].esal)

		fmt.Println("-------------------- ***** ------------------")
	}
	max = e1[0].esal
	for i := 0; i < n; i++ {
		if e1[i].esal>max {
			max = e1[i].esal
			k=i
```

```
            }
        }
        fmt.Println("-------------------- ***** ------------------")
        fmt.Println("Employee Having maximum Salary ...")
        fmt.Println("Emloyee No : ",e1[k].eno)
        fmt.Println("Emloyee Name : ",e1[k].ename)
        fmt.Println("Emloyee Salary : ",e1[k].esal)

}
```

3. WAP in go language to demonstrate working of slices (like append, copy etc.)

```
package main
import "fmt"

func main(){
        str :=[]string{
                "Mango",
                "Apple",
                "Banana",
        }
        fmt.Println(str)
        a := make([]string, len(str))

        s1 :=append(str,"Pinepple")
        fmt.Println(s1)

        fmt.Println(copy(a, str))
        fmt.Printf("%v",a)

}
```

Assignment 4

SETA

1. 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"

//Interface declaration
type shape interface {
   area() float64
   perimeter() float64
}
//Struct declaration for rectangle
type rectangle struct{
  length, height float64
}

//Struct declaration for circle
type circle struct{
  radius float64
}

//Method declarations for rectangle
func (r rectangle) area() float64 {
   return r.length * r.height
}
func (r rectangle) perimeter() float64 {
   return 2 * r.length + 2 * r.height
}
//Method declarations for circle
func (c circle) area() float64 {
   return 3.142 * c.radius * c.radius
```

```
}
func (c circle) perimeter() float64 {
   return 2 * 3.142 * c.radius
}

func main() {
   r := rectangle{length: 10.0, height: 5.0}
   c := circle{radius: 5.0}

   fmt.Println("Area of rectangle is ", r.area())
   fmt.Println("Parameter of rectangle is ", r.perimeter())
   fmt.Println("Area of circle is ", c.area())
   fmt.Println("Perimeter of circle is ", c.perimeter())

}
```

2. Write a program in go language to print multiplication of two numbers using method

```
package main
import "fmt"

type data int

func (d1 data) multipy (d2 data) data {
   mult := d1 * d2
   return mult
}

func main() {
   value1 := data(23)
   value2 := data(20)

   res := value1.multipy(value2)
   fmt.Println("Final Result : " ,res)
}
```

3. Write a program in go language to create structure author. Write a method show() whose receiver is struct author.

```
package main
import "fmt"

type author struct{
	branch,name string
	particles,salary int
}

func (a author) show(){
	fmt.Println("Author's Name : ", a.name)
	fmt.Println("Branch Name : ", a.branch)
	fmt.Println("Published Articles : ", a.particles)
	fmt.Println("Salary : ", a.salary)
}

func main(){
	res := author{
		name : "Tejas",
		branch : "BCA",
		particles : 03,
		salary: 734000,
	}

	res.show()
}
```

SETB

1. 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
	roll int
}

func (s *student) show (sroll int,sname string){
	(*s).roll = sroll
	(*s).name = sname
}

func main() {
	res := student{
		name : "Ashish",
		roll: 1,
	}
	fmt.Println("Roll No : ", res.roll)
	fmt.Println("Student Name : ",res.name)

	p := &res

	p.show(2,"Tejas")
	fmt.Println("Roll No : ", res.roll)
	fmt.Println("Student Name : ",res.name)


}
```

2. Write a program in go language to demonstrate working type switch in interface.

```
package main

import ("fmt")

func main() {
```

```
        var value interface{} = "2"
        switch t := value.(type){
            case int64:
                fmt.Println("Type is an integer:", t)
            case float64:
                fmt.Println("Type is a float:", t)
            case string:
                fmt.Println("Type is a string:", t)
            case nil:
                fmt.Println("Type is nil.")
            case bool:
                fmt.Println("Type is a bool:", t)
            default:
                fmt.Println("Type is unknown!")
        }
    }
```

SETC

1. Write a program in go language to create an interface and display it's values with the help of type assertion.

```
package main
import "fmt"
func main(){
    var myInt interface{} = 123
    k, ok := myInt.(int)
    if ok {
        fmt.Println("Success :", k)
```

```
        }

        v, ok := myInt.(float64)
        if ok {
            fmt.Println(v)
        }else{
            fmt.Println("Failed Without Panicking !")
        }
    }
```

2. Write a program in go language to demonstrate working embedded interfaces.

```
package main

import "fmt"

// Interface 1

type AuthorDetails interface {

details()

}

// Interface 2

type AuthorArticles interface {

articles()

}

// Interface 3

// Interface 3 embedded with

// interface 1 and 2

type FinalDetails interface {

    AuthorDetails

    AuthorArticles
```

```
}
// Structure
type author struct {
        a_name string
        branch string
        college string
        year int
        salary int
        particles int
        tarticles int
}
// Implementing method of
// the interface 1
func (a author) details() {
        fmt.Printf("Author Name: %s", a.a_name)
        fmt.Printf("\nBranch: %s and passing year: %d",
        a.branch, a.year)
        fmt.Printf("\nCollege Name: %s", a.college)
        fmt.Printf("\nSalary: %d", a.salary)
        fmt.Printf("\nPublished articles: %d", a.particles)
}
// Implementing method of the interface 2
func (a author) articles() {
```

```
        pendingarticles := a.tarticles - a.particles
        fmt.Printf("\nPending articles: %d", pendingarticles)
}
        // Main value
func main() {
        // Assigning values
        // to the structure
        values := author{
        a_name: "Shirwaikar",
        branch: "Computer science",
        college: "XYZ",
        year: 1990,
        salary: 80000,
        particles: 209,
        tarticles: 309,
        }
        // Accessing the methods of
        // the interface 1 and 2
        // Using FinalDetails interface
        var f FinalDetails = values
        f.details()
        f.articles()
}
```

Assignment 5

SETA

2. WAP in GO program that executes 5 go routines simultaneously which generates numbers from 0 to 10, waiting between 0 and 250 ms after each go routine.

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

func numbers() {
   for i := 1; i <= 5; i++ {
      time.Sleep(250 * time.Millisecond)
      fmt.Printf("%d ", i)
   }
}
func alphabets() {
   for i := '0'; i <= '10'; i++ {
      time.Sleep(400 * time.Millisecond)
      fmt.Printf("%c ", i)
   }
}
func main() {
   go numbers()
   go alphabets()
   time.Sleep(3000 * time.Millisecond)
   fmt.Println("main terminated")
}
```

3. 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() {
	var intSlice = []int{91, 42, 23, 14, 15, 76, 87, 28, 19, 95}
	chOdd := make(chan int)
	chEven := make(chan int)
	go odd(chOdd)
	go even(chEven)
	for _, value := range intSlice {
		if value%2 != 0 {
			chOdd <- value
		} else {
			chEven <- value
		}
	}
}
func odd(ch <-chan int) {
	for v := range ch {
		fmt.Println("ODD :", v)
	}
}
func even(ch <-chan int) {
	for v := range ch {
		fmt.Println("EVEN:", v)
```

```
        }

    }
```

SETB

1. WAP in Go 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() {
        // create a buffered channel
        // with a capacity of 2.
        ch := make(chan string, 2)
        ch <- "geeksforgeeks"
        ch <- "geeksforgeeks world"
        fmt.Println(<-ch)
        fmt.Println(<-ch)
        fmt.Println("Capacity of Buffered Channel : ",cap(ch))
        fmt.Println("Capacity of Buffered Channel : ",len(ch))
}
```

2. WAP in Go main go routine to read and write Fibonacci series to the channel

```
package main


import (

        "fmt"

)
```

```
func fibonacci(ch chan int, quit chan bool) {
	x, y := 0, 1
	for {
		select {
		case ch <- x: // write to channel ch
			x, y = y, x+y
		case <-quit:
			fmt.Println("quit")
			return
		}
	}
}

func main() {
	ch := make(chan int)
	quit := make(chan bool)
	n := 10

	go func(n int) {
		for i := 0; i < n; i++ {
			fmt.Println(<-ch) // read from channel ch
		}
		quit <- false
```

```
        }(n)


        fibonacci(ch, quit)
    }}
```

3. WAP in Go how to create channel and illustrate how to close a channel using for range loop and close function.

```
        package main


        import (
                "fmt"
                "sync"
        )


        var wg sync.WaitGroup = sync.WaitGroup{}


        func main() {
                ch := make(chan int, 50)
                wg.Add(2)
                go func(ch <-chan int) {
                        for {
                                if i, ok := <-ch; ok {
                                        fmt.Println(i)
                                } else {
                                        break
                                }
```

```
        }
        wg.Done()
    }(ch)
    go func(ch chan<- int) {
        var i int
        i = 17
        ch <- i
        ch <- i + 18
        ch <- 13
        ch <- 19
        close(ch)
        wg.Done()
    }(ch)
    wg.Wait()
}
```