Some of the important statements in Go programming

- ✓ **Package main**

The package main tells the Go compiler that the package should compile as an executable program instead of a shared library.

In other words, The shared library are the one which do some task that is commonly accessed or used by executables. These library are loaded into the memory only once and accessed by many programs(executables) at runtime.

- ✓ **Import "fmt"**

A package in Go programming which is most commonly used is fmt package , where fmt stands for format package. This package allows to format basic strings, values, or anything and print them or collect user input from console, or write into a file using a writer or even print customized fancy error messages. This package is all about formatting input and output.

- ✓ **Func main()**

The main() function is a special type of function and it is an entry point of the executable programs. It does not take any arguments nor return anything.

- ✓ **Function declaration**

A function is a block of code that takes some input, does some processing on the input and produces some output.

Function helps divide your program into small reusable pieces of code. They improve the readability, maintainability, and testability of the program.

We declare a function using the func keyword. A function has a name, a list of comma-separated input parameters along with their types, the result type, and a body.

Syntax: func functionName(Parameters){

Body}

- ✓ **Variable declaration**

A variable is a piece of storage containing data temporarily to work with it.

Syntax: var name type = expression

✓ **fmt.Scanln() Function**

The fmt.Scanln() function in Go programming scans the input texts which is given in the standard input, reads from there stored the successive space separated values into successive arguments. This function stops scanning at a newline and after the final item, there must be a newline or EOF.

✓ **fmt.Println() function**

The fmt.Println() function in Go programming formats using the default formats for its operands and writes to standard output. Here spaces are always added between operands and a newline is appended at the end. Moreover, this function is defined under the fmt package.

✓ **'for' loop syntax:**

```
for initialization; condition; post{
    // statements....
}
```

✓ **Binary Search Algorithm:**

The basic steps to perform Binary Search are:

- Begin with the mid element of the whole array as search key.
- If the value of the search key is equal to the item then return index of the search key.
- Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
- Otherwise, narrow it to the upper half.
- Repeatedly check from the second point until the value is found or the interval is empty.

✓ **Linear Search Algorithm:**

A simple approach is to do a **linear search**, i.e

- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]

- If x matches with an element, return the index.
- If x doesn't match with any of elements, return -1.

✓ For-range loop

A **for loop** is used to iterate over elements in a variety of data structures (e.g., a slice, an array, a map, or a string).

The **for statement** supports one additional form that uses the keyword range to iterate over an expression that evaluates to an array, slice, map, string, or channel.

✓ Interface

Go language interfaces are different from other languages. In Go language, the interface is a custom type that is used to specify a set of one or more method signatures and the interface is abstract, so you are not allowed to create an instance of the interface. But you are allowed to create a variable of an interface type and this variable can be assigned with a concrete type value that has the methods the interface requires. Or in other words, the interface is a collection of methods as well as it is a custom type.

Syntax:type interface_name interface{
// Method signatures
}

✓ Structures

A structure or struct in Golang is a user-defined type that allows to group/combine items of possibly different types into a single type. Any real-world entity which has some set of properties/fields can be represented as a struct. This concept is generally compared with the classes in object-oriented programming. It can be termed as a lightweight class that does not support inheritance but supports composition.

1. GO Program to Find LCM and GCD of given two numbers.

```
package main

import "fmt"

func lcm(temp1 int, temp2 int) { // Function is declared with a name ‘lcm’ and
//parameters temp1 and temp2 of integer type

var lcmnum int = 1// variable is declared with a name ‘lcmnum’ of integer type and
initialized with ‘1’ as a value.


//Inorder to find the number which is greatest between two numbers the following
//condition statement (if-else) is used.

if temp1 > temp2 {

lcmnum = temp1//if the first argument is greater than the second , then ‘lcmnum’ will
//contain the first argument.

} else {

lcmnum = temp2//if the second argument is greater than the first, then ‘lcmnum’ will
//contain the second argument.
}

for {
if lcmnum%temp1 == 0 && lcmnum%temp2 == 0 {
//if the number is divisible by both the arguments then that number will be considered
//as the lcm of both the numbers.


fmt.Printf("LCM of %d and %d is %d", temp1, temp2, lcmnum)//printing statement
//Here %d is a format specifier and hence specifies that the type of variable as decimal


break//if the above condition is satisfied then the loop ends, and this work is done ny
//break statement
}

lcmnum++//incrementing ‘lcmnum’
}

return//returns the final output of ‘lcm’ function
}

func gcd(temp1 int, temp2 int) {// Function is declared with a name ‘gcd’ and
//parameters temp1 and temp2 of integer type
```

```
var gcdnum int // variable is declared with a name ‘lcmnum’ of integer type.
//initializing the variable ‘gcdnum’ or not initializing doesnt show any effect in this
//program. Both the cases works the same.



for i := 1; i <= temp1 && i <= temp2; i++ {//for loop in go programming

if temp1%i == 0 && temp2%i == 0 {
//the number which divides both the arguments will be considered as their GCD,
//Inorder to find the same we will use the above if condition.


gcdnum = i//The number which is divides both the number is stored inside the
//variable ‘gcdnum’
}
}

fmt.Printf("GCD of %d and %d is %d", temp1, temp2, gcdnum)//printing statement


return //returns the final output of ‘gcd’ function
//note with or without return statement in this program doesnt effect much. The
//program works the same in both the cases.

}

func main() {//main function

var n1, n2, action int //initializing variables with name n1 , n2 ,action of integer type.

fmt.Println("Enter two positive integers : ")//printing satement which tells user to
//enter two positive integers

fmt.Scanln(&n1)//scans the input texts from the user

fmt.Scanln(&n2)//scans the input texts from the user


fmt.Println("Enter 1 for LCM and 2 for GCD")//printing satement which tells user to
//enter the choice , that is whether user want to find LCM or GCD


fmt.Scanln(&action)//scans the input texts from the user


switch action {//switch statement for ‘action’ variable where the choice that the user
//specified by the user is stored
```

```
case 1://if the choice is '1' then 'lcm' function is called and the output is printed
//accordingly.
lcm(n1, n2)

case 2://if the choice is '2' then 'gcd' function is called and the output is printed
//accordingly.
gcd(n1, n2)
}
}
```

OUPUT:

```
PS G:\Desktop\Go Lang\lab> go run gcdlcm.go
Enter two positive integers :
5
10
Enter 1 for LCM and 2 for GCD
1
LCM of 5 and 10 is 10
PS G:\Desktop\Go Lang\lab> 2
2
PS G:\Desktop\Go Lang\lab> go run gcdlcm.go
Enter two positive integers :
4
10
Enter 1 for LCM and 2 for GCD
2
GCD of 4 and 10 is 2
PS G:\Desktop\Go Lang\lab>
```

Requirement

How to find LCM of two numbers?

- **Finding LCM of Two Numbers by Listing Multiples Method**

In the listing multiples method, we have to write down the multiples of two numbers up to the first common multiple. For example, find LCM of 5 and 9. Multiples of number 5 = 5, 10,15, 20, 25, 30, 35, 40, 45 and so on. Multiples of number 9 = 9, 18, 27, 36, 45 and so on. The first smallest common multiple is 45, of numbers 5 and 9. Hence the LCM of the two numbers is 45.

- **Finding LCM of Two Numbers by Prime Factorization Method**

In this method, first, write down all the prime factors of two numbers then multiply each factor with the highest powers to calculate the LCM of two numbers. For example, find LCM of 18 and 24.

Prime factors of number 18 = 2×3^2

Prime factors of number 24 = $2^3 \times 3$

LCM of numbers 18 and 24 = $2^3 \times 3^2 = 72$

Here we multiply all the prime factors that are 2 and 3 with the highest powers, to find the LCM of two numbers.

- **Finding LCM of two numbers by Division Method**

To find LCM of two numbers by division method follow the following steps given below:

- Step 1: Write the two numbers in a horizontal line and separate the numbers by commas.
- Step 2: Divide the two numbers with prime numbers, if any number is not exactly divided write down the number in the next row.
- Step 3: Keep on dividing the two numbers by prime numbers, until not getting the results as 1 in the complete row.
- Step 4: LCM of two numbers is the product of all prime numbers obtained in a division method.

For example, let's find LCM of 6 and 15 by division method.

- Write the numbers 6 and 15 in a horizontal line and separate them by commas.
- Divide the 6 and 15 by prime numbers, if a number is not exactly divided write down in the next row.
- Divide the numbers 6 and 15 till not getting 1 in a complete row.
- LCM of 6 and 15 is the product of all prime numbers i.e. $2 \times 3 \times 5 = 30$

How to find GCD of two numbers?

For a set of two positive integers (a, b) we use the below-given steps to find the greatest common divisor:

- **Step 1:** Write the divisors of positive integer "a".
- **Step 2:** Write the divisors of positive integer "b".
- **Step 3:** Enlist the common divisors of "a" and "b".

- **Step 4:** Now find the divisor which is the highest of both "a" and "b".

Example: Find the greatest common divisor of 13 and 48.

Solution: We will use the below steps to determine the greatest common divisor of (13, 48).

Divisors of 13 are 1, and 13.

Divisors of 48 are 1, 2, 3, 4, 6, 8, 12, 16, 24 and 48.

The common divisor of 13 and 48 is 1.

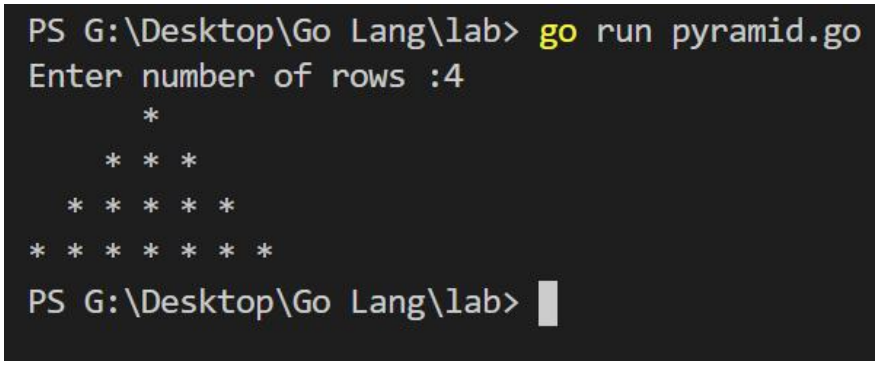
The greatest common divisor of 13 and 48 is 1.

Thus, GCD(13, 48) = 1.

2. GO Program to print full Pyramid using STAR.

```
package main//package declaration
import "fmt" //fromat package
func main() {//main function

var i, j, rows int //variable declaration of name ‘i’, ‘j’, and ‘rows’ of integer type

fmt.Println("Enter the number of rows")//Printing statement to ask the user to print the
//number of rows the pyramid should contain.

fmt.Scanln(&rows)//scans the input texts from the user and stores the value inside
//‘rows’ variable

for i = 1; i <= rows; i++ {//This ‘for’ condition is used to make sure that the total
//number of rows of the pyramid must not exceed the value entered by the user
for j = 1; j <= rows-i; j++ {//This ‘for’ condition is used to print ‘nothing/space’ in
//horizontal order , so that the shape of the pyramid is obtained.
fmt.Print(" ")//Printing statement that prints ‘nothing/space’ in horizontal order when
the //above for condition is satisfied

}
for k := 0; k != (2*i - 1); k++ {//This ‘for’ condition is used to print ‘*’ according to
//the condition.
//in the above ‘for’ loop the value of ‘k’ is initialized to 0 (k:=0 is short hand
//representation for declaring a variable with name ‘k’ , type ‘int’ and initial value ‘0’
//(2*i-1) is solved according to BODMAS technique , so multiplication is done before
//subtraction.
fmt.Print("*")//printing statement that prints ‘*’
}
fmt.Println()//Printing statement that takes you to the next horizontal line
}
}
```

OUPUT:

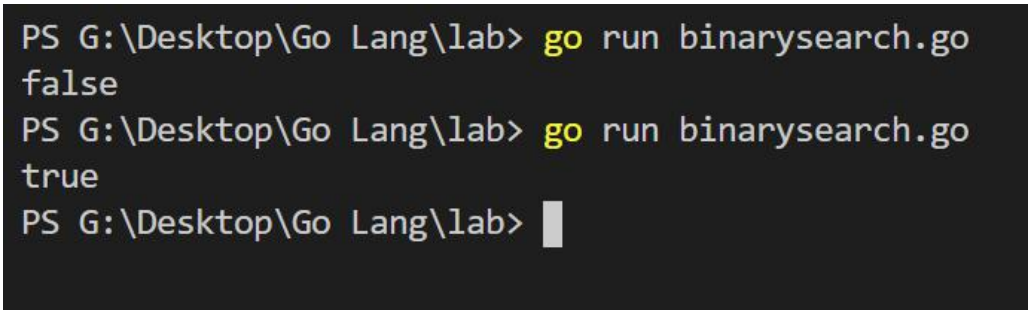

PS G:\Desktop\Go Lang\lab> go run pyramid.go
Enter number of rows :4
*
* * *
* * * * *
* * * * * * *
PS G:\Desktop\Go Lang\lab>

3. GO program for implementation of Binary Search.

```
package main //package declaration
import "fmt" //format package

func binarySearch(needle int, haystack []int) bool { //function with name
//'binarySearch' with parameters needle (int type), haystack (array) is declared
//here 'bool' represents the return type of the function binarySearch

low := 0              //short hand representation of a variable with a name 'low' , and
//initial value 0 is declared

high := len(haystack) //short hand representation of a variable with a name 'high' , and
//initial value contains the total length of the array when the binary search takes place
//is declared

for low <= high {     //'for' loop is used, inorder to see whether the starting index value
//of an element is less than or equal to the last index value of an element
//this 'for' loop is just to make sure that there are some elements present in an array

median := (low + high) / 2     //a variable with name 'median' is declared and it stores
//the value which is obtained by solving (low+high)/2 (i.e, the index value of an
//element which is present in the middle of an array)

if haystack[median] < needle { //If the element which is considered as the middle
//element in an array is less than the element to be searched , then the variable 'low'
//will store the value ( median + 1), which means all the element before this index
//number will be neglected.

low = median + 1

} else { //if 'if' condition is not satisfied then else condition takes action, henceforth
//changing the value of the variable 'high' to (median-1), such that the elements after
//this index number will be neglected.
```

```
high = median - 1
}
}
if low == len(haystack) || haystack[low] != needle { //after minimizing an array
//according to the binary search method, now we need to search if the element which
//is being searched is present in the modified array or not.
// if the above 'if'statement is satisfied then it means the element which is being
searched is not present in an array and hence returns 'false'

return false
}
//if the above 'if' condition is not satisfied then it means the element which is being
//searched is present in an array and hence returns 'true'

return true
}
func main() { //main function
items := []int{1, 2, 9, 20, 31, 45, 63, 70, 100} //an array of name 'items' ,datatype int ,
//is declared and stores an array
fmt.Println(binarySearch(101, items))
//fmt.Println(binarySearch(100, items))
//a function 'binarySearch' inside the printing //statement is called where the first
parameter represents the element to be searched //and the second parameter represents
the array where the element need to be searched.
}
```

OUPUT:

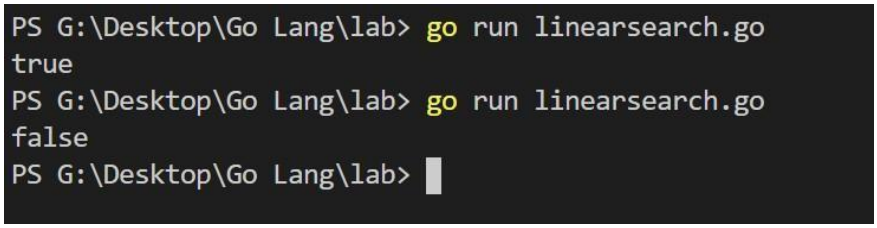


4. GO program for implementation of Linear Search.

```
package main //package declaration
import "fmt" //format package

func linearSearch(datalist []int, key int) bool { //function with name
//'linearSearch' with parameters key (int type), datalist (array) is declared.
//here 'bool' represents the return type of the function linearSearch

for _, item := range datalist { //for-range loop ,when you are not interested in some
//values returned by a function we use underscore in place of variable name. Basically,
//it means you are not interested in how many times the loop has run till now just that
//it should run some specified number of times overall.

//keyword 'range' to iterate over an expression that evaluates to an array, slice, map,
string, or channel.

if item == key { //if the item matches the 'key' value then it returns 'true' or else the
//loop runs again until all the elements in an array is considered.

return true
}
}
//if the above 'if' statement is not satisfied then it returns 'false'

return false
}
func main() { //main function

items := []int{95, 78, 56, 84, 25, 35, 15, 26} //an array of name 'items' ,datatype int ,
//is declared and stores an array
```

```
fmt.Println(linearSearch(items, 95))

//fmt.Println(linearSearch(items, 96))
//a function 'binarySearch' inside the printing
//statement is called where the first parameter represents the element to be searched
//and the second parameter represents the array where the element need to be searched.
}
```

OUPUT:



5. GO Program to Generate Multiplication Table.

```
package main //package declaration
import "fmt" //format package

func main() { //main function

var n int      //A variable of name 'n' and type 'int' is declared

fmt.Print("enter the integer number :") //Printing statement which asks user to print
//the number, of which the multiplication table is obtained.

fmt.Scan(&n) //scans the input texts from the user and stores the value inside 'n'
//variable

i := 1  //a variable of name 'i' and type 'int' and initialized with a value '1' is declared

for { //here 'for' loop is used as an infinite loop by removing all the three expressions
//from the for loop. When the user did not write condition statement in for loop it
//means the condition statement is true and the loop goes into an infinite loop.

if i > 10 { //the multiplication table is printed until the multiplication of 10 so the
//condition is written as followed

break //if 'i' becomes greater than 10 then the loop breaks
}

fmt.Println(n, "X", i, "=", n*i) //if 'i' is not greater than 10 , it gets multiplied with the
//number of which the multiplication table needs to be created.

i++   // 'i' value gets iterated
}
}
```

OUPUT:

```
PS G:\Desktop\Go Lang\lab> go run multiplicationtable.go
enter the integer number :4
4 X 1 = 4
4 X 2 = 8
4 X 3 = 12
4 X 4 = 16
4 X 5 = 20
4 X 6 = 24
4 X 7 = 28
4 X 8 = 32
4 X 9 = 36
4 X 10 = 40
PS G:\Desktop\Go Lang\lab>
```

6. GO Program to Add Two Matrix Using Multi-dimensional Arrays.

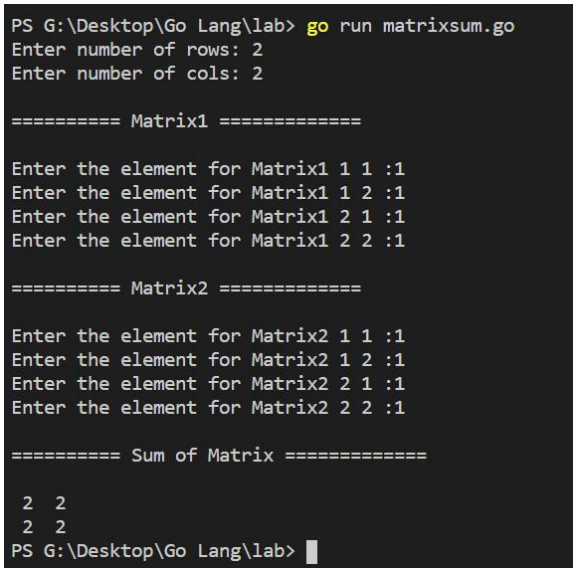
```
package main //package declaration
import "fmt" //format package

func main() { // main function

var matrix1 [100][100]int   //here an array of name 'matrix1' which contains row of
//length 100, and column of length 100 to be the limit and type integer is declared

var matrix2 [100][100]int  //here an array of name 'matrix2' which contains row of
//length 100, and column of length 100 to be the limit and type integer is declared

var sum [100][100]int //here an array of name 'sum' which contains row of length 100,
//and column of length 100 to be the limit and type integer is declared

var row, col int  // variables of name 'row', 'col' and type integer is declared

fmt.Println("enter number of rows :")//printing statement that asks user to print the
//number of rows the matrix contains

fmt.Scanln(&row)   //scans the input texts from the user and stores the value inside
//'row' variable

fmt.Println("enter number of cols :") //printing statement that asks user to print the
//number of columns the matrix contains

fmt.Scanln(&col)   //scans the input texts from the user and stores the value inside
//'col' variable

fmt.Println()  //printing statement to move into next line

fmt.Println("==================matrix1=================")
//printing statement

fmt.Println() //printing statement to move into next line
```

```
for i := 0; i < row; i++ {    //variable 'i' is declared of integer type and initial value to
//be '0'

for j := 0; j < col; j++ {     //variable 'j' is declared of integer type and initial value to
//be '0'

//the above two 'for' loops are used to let user input value to each position of the
//matrix(first matrix) which depends on the number of rows and columns entered by
//the user, and hence runs until all the elements are obtained from the user.

fmt.Printf("enter the element for matrix1 %d %d :", i+1, j+1) //printing statement
//which tells the position of the element where the user needs to input the value

fmt.Scanln(&matrix1[i][j])   //scans the input texts from the user and stores the value
//inside 'matrix1' array accorind to the position (i and j values).

}
}

fmt.Println() //printing statement to move into next line

fmt.Println("==================matrix2==================")
//printing statement

fmt.Println()  //printing statement to move into next line

for i := 0; i < row; i++ {    //variable 'i' is declared of integer type and initial value to
//be '0'

for j := 0; j < col; j++ {     //variable 'j' is declared of integer type and initial value to
//be '0'

//the above two 'for' loops are used to let user input value to each position of the
//matrix(second matrix) which depends on the number of rows and columns entered
//by the user, and hence runs until all the elements are obtained from the user.

fmt.Printf("enter the element for matrix2 %d %d :", i+1, j+1)   //printing statement
//which tells the position of the element where the user needs to input the value
```

```
fmt.Scanln(&matrix2[i][j])   //scans the input texts from the user and stores the value
//inside 'matrix2' array accorind to the position (i and j values).
}
}

for i := 0; i < row; i++ {     //variable 'i' is declared of integer type and initial value to
//be '0'

for j := 0; j < col; j++ {     //variable 'j' is declared of integer type and initial value to
//be '0'

//the above two 'for' loops are used to add an element from both the matrix having the
//same position, and hence loop repeats itself until all the elements are added.

sum[i][j] = matrix1[i][j] + matrix2[i][j]//To add both the elements that are present in
//the same position but different matrix
}
}
fmt.Println()//printing statement to move into next line

fmt.Println("========== Sum of Matrix ==========") //printing statement

fmt.Println() //printing statement to move into next line

for i := 0; i < row; i++ {     //variable 'i' is declared of integer type and initial value to
//be '0'

for j := 0; j < col; j++ {     //variable 'j' is declared of integer type and initial value to
//be '0'

//the above two 'for' loops are used to display the result after adding both matrix1 and
//matrix2 in matrix format.

fmt.Printf(" %d ", sum[i][j]) //printing statement which is used to display the sum of
//elements from both the matrix that are present at the same position in their
//respective matrix.

if j == col-1 { // 'if' statement is used inorder to obtain space after each element(sum
//product) is printed, making the output more presentable and easier to understand
```

```
fmt.Println("") //printing statement to print the space
}
}
}
}
```

OUPUT:

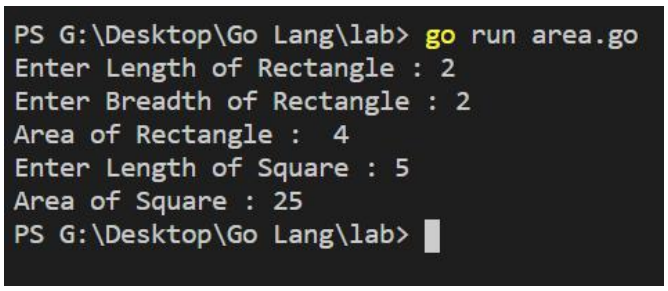


7. GO Program to Calculate Area of Rectangle and Square.

```
package main //package declaration
import "fmt" //format package

var area int //declaring a variable with name 'area' of 'int' type.

func main() { //main function

var l, b int  //declaring variables with name 'l', 'b' of 'int' type.

fmt.Println("enter length of rectangle :")   //printing statement which tells user to enter
//the length of a rectangle

fmt.Scanln(&l)   //scans the input texts from the user and stores the value inside 'l'

fmt.Println("enter the breadth of rectangle :") //printing statement which tells user to
//enter the bredth of a rectangle

fmt.Scanln(&b) //scans the input texts from the user and stores the value inside 'b'

area = l * b //applying the formula to calculate area of rectangle(i.e , length*bredth)

fmt.Println("area of rectangle :", area)  //printing statement which gives the calculated
//output (area of the rectangle)

fmt.Println("enter length of square :")  //printing statement which tells user to enter
//the length of a square

fmt.Scanln(&l)//scans the input texts from the user and stores the value inside 'l'

area = l * l  //applying the formula to calculate area of rectangle(i.e , length*length)
```

```
fmt.Println("area of square :", area)   //printing statement which gives the calculated
//output (area of the square)
}
```

OUPUT:



8. GO Program to Check Whether a Number is Palindrome or Not.

```
package main //package declaration
import "fmt" //format package

func main() { //main function

var number, remainder, temp int   //declaring variables of name 'number', 'remainder',
//'temp' and type 'int'.

var reverse int = 0   //declaring a variable of name 'reverse' and type 'int' and initial
//value '0'.

fmt.Println("enter any positive integer :") //printing statement which tells user to enter
//the positive integer.

fmt.Scanln(&number)  //scans the input texts from the user and stores the value inside
//'number'.

temp = number  //variable 'temp' which contains the same value as that of 'number'
//variable.

for { //here 'for' loop is used as an infinite loop by removing all the three expressions
//from the for loop. When the user did not write condition statement in for loop it
//means the condition statement is true and the loop goes into an infinite loop.

remainder = number % 10
reverse = reverse*10 + remainder
number /= 10
```

```
//the above three statement is used to reverse the positive number entered by the user
//(the above 'for' loop is repeated until the number is completely reversed), inorder to
//check whether the reversed number is same as that of the original number.

if number == 0 {   //if the 'number' is excatly equal to '0' then the 'for' loop breaks
break
}
}
if temp == reverse { //if the original number given by the user is exactly equal to its
//reversed form then the number is known as a palindromic number

fmt.Println("%d is a palindrome", temp) //printing statement

} else { //if the original number given by the user is not equal to its reversed form then
//the number is not a palindromic number

fmt.Println("%d is not a palindrome", temp) //printing statement
}
}
```

OUPUT:

```
PS G:\Desktop\Go Lang\lab> go run palindrome.go
Enter any positive integer : 43
43 is not a Palindrome
PS G:\Desktop\Go Lang\lab> go run palindrome.go
Enter any positive integer : 22
22 is a Palindrome
PS G:\Desktop\Go Lang\lab>
```

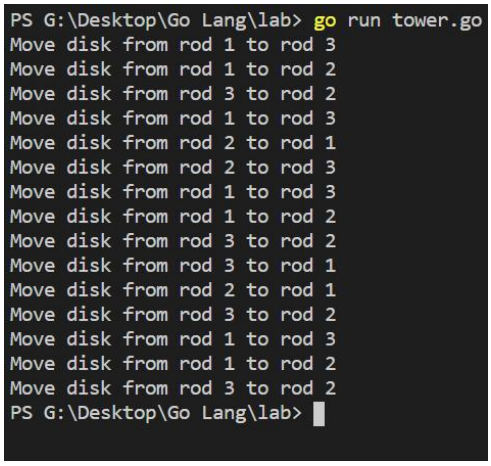
9. GO program to implementation of Tower of Hanoi Algorithm.

```
package main //package declaration
import "fmt" //format package

type solver interface { //interface in go programming

play(int) // a method named 'play' and contains a parameter of integer type is //called
}

type towers struct { //A struct in Go language is a yser-defined type that allows to
//group/combine items of possibly different types into a single type. Any real-world
//entity which has some set of properties/fields can be represented as a struct.
}


func (t *towers) play(n int) { //here struct which is declared above is passed into the
//function. It means the whole structure is passed to another function with all members
//and their values. So, this structure can be accessed from called function
//the above function returns the value 'play(n int)' method

t.moveN(n, 1, 2, 3) //'moveN' method is called by 't' that refers to 'tower' struct.
}

func (t *towers) moveN(n, from, to, via int) { //the above function returns the value
//'moveN(n, from, to, via int)' method

if n > 0 { //if the number of blocks is greater than '0' then the blocks are moved

t.moveN(n-1, from, via, to) //here 'moveN' method is called from 't' //that points to
'towers' struct via pointers

//here the first parameter is the number of blocks that need to be //moved , second
parameter represents the rod from which the disk needs to be //moved , third
```

```
parameter represents the rod through which the disk are moved , fourth //parameter
represents the destined rod to which the disk are moved.

t.moveM(from, to) //here 'moveM' method is called from 't' that points //to 'towers'
struct via pointers

//here first parameter represents the rod from which the disk needs to be moved ,
//second parameter represents the destined rod to which the disk are moved.

t.moveN(n-1, via, to, from) //here 'moveN' method is called from 't' that points to
//'towers' struct via pointers

//here the first parameter is the number of blocks that need to be moved , second
//parameter represents the rod from which the disk needs to be moved , third
//parameter represents the rod through which the disk are moved , fourth parameter
//represents the destined rod to which the disk are moved.
}
}

func (t *towers) moveM(from, to int) {
//the above function returns the value 'moveM(from, to int)' method

fmt.Println("move disk from rod", from, "to rod", to) //printing statement
}

func main() { //main function

var t solver    //variable of interface type which can hold any value implementing
//particular interface.

t = new(towers) //An instance of a struct can also be created with the 'new' keyword. It
//is then possible to assign data values to the data fields using dot notation.
```

```
t.play(4)
}
```

OUPUT:



10. GO Program to read file line by line to string.

```
package main//package declaration

import (

"bufio"//bufio package is used for buffered IO, Buffereing IO is a technique used to
//temperorily accumulate the results for an IO operation before transmitting it
//forward.This technique can increase the speed of the program by reducing the
//number of system calls, which are typically slow operations.

"fmt"//format package

"log"//The package log in go language implements the simple logging package. It
//defines a type, logger, with methods for formatting output. Golang Log will be
//helpful in the critical scenerios in real-time applications.

"os"//Package os provides a platform-independent interface to operating system
//functionality. The design is Unix-like, although the error handling is Go-like; failing
//calls return values of type error rather than error numbers. Often, more information
//is available within the error.
)

func main() {//main function

file, err := os.Open("kotlin.txt")//This step is used to open a file (kotlin.txt) for
//reading. We can use the os package Open() function to open the file.

if err != nil {//if the file which is read is empty then the statement within 'if' condition
//is printed.

log.Fatalf("failed opening file: %s", err)//Inorder to print specified message with
//timestamp on the console screen the Fatalf function from 'log' package is used
```

```
}
scanner := bufio.NewScanner(file)  //to read the file line by line 'NewScanner' method
//of 'bufio' package is used , and the value obtained will be stored into a variable
//'scanner'

scanner.Split(bufio.ScanLines)  //The string method in golang breaks down the string
//down into list of substrings using a specified separator. The method returns the
//substrings in the form of a slice.

//The bufio.ScanLines is used as an input to the method bufio.Scanner.Split() and then
//the scanning forwards to each new line using the bufio.Scanner.Scan() method.
//var txtlines []string//array of name 'txtlines' and type string

for scanner.Scan() {

txtlines = append(txtlines, scanner.Text())  //The built-in append function appends
//elements to the end of a slice: if there is enough capacity, the underlying array is
//reused ; if not , a new underlying array is allocated and the data is copied over.

}

file.Close()//closing the file which is being read

for _, eachline := range txtlines {//for range loop

fmt.Println(eachline)      //printing statement for printing each and every line of the
//text file
}
}
```

OUPUT:

```
PS G:\Desktop\Go Lang> cd lab
PS G:\Desktop\Go Lang\lab> go run readline.go
hello world
PS G:\Desktop\Go Lang\lab>
```

11. **GO Program to Get current date and time in various format in golang.**

```
package main //package declaration

import (
"fmt"  //format package
"time" //package for time and durations.
)

func main() { //main function

currentTime := time.Now() //variable 'currentTime' which stores current time using
//Now() method.

fmt.Println("Current Time in String: ", currentTime.String()) //to obtain current time
//in string format we use String() method from currentTime class.

fmt.Println("MM-DD-YYYY : ", currentTime.Format("01-02-2006"))//'format'
//method specifies the format in which the output needs to be printed.

fmt.Println("YYYY-MM-DD : ", currentTime.Format("2006-01-02"))
fmt.Println("YYYY.MM.DD : ", currentTime.Format("2006.01.02 15:04:05"))
fmt.Println("YYYY#MM#DD {Special Character} :
",currentTime.Format("2006#01#02"))
fmt.Println("YYYY-MM-DD hh:mm:ss : ", currentTime.Format("2006-01-02
15:04:05"))
fmt.Println("Time with MicroSeconds: ", currentTime.Format("2006-01-02
15:04:05.000000"))
fmt.Println("Time with NanoSeconds: ", currentTime.Format("2006-01-02
15:04:05.000000000"))
fmt.Println("ShortNum Month : ", currentTime.Format("2006-1-02"))
fmt.Println("LongMonth : ", currentTime.Format("2006-January-02"))
```

```
fmt.Println("ShortMonth : ", currentTime.Format("2006-Jan-02"))
fmt.Println("ShortYear : ", currentTime.Format("06-Jan-02"))
fmt.Println("LongWeekDay : ", currentTime.Format("2006-01-02 15:04:05 Monday"))
fmt.Println("ShortWeek Day : ", currentTime.Format("2006-01-02 Mon"))
fmt.Println("ShortDay : ", currentTime.Format("Mon 2006-01-2"))
fmt.Println("Short Hour Minute Second: ", currentTime.Format("2006-01-02 3:4:5"))
fmt.Println("Short Hour Minute Second: ", currentTime.Format("2006-01-02 3:4:5
PM"))
fmt.Println("Short Hour Minute Second: ", currentTime.Format("2006-01-02 3:4:5
pm"))
}
```

OUPUT:

```
PS G:\Desktop\Go Lang\lab> go run date.go
Current Time in String:  2021-04-13 11:18:31.6352819 +0530 IST m=+0.003619401
MM-DD-YYYY :  04-13-2021
YYYY-MM-DD :  2021-04-13
YYYY.MM.DD :  2021.04.13 11:18:31
YYYY#MM#DD {Special Character} :  2021#04#13
YYYY-MM-DD hh:mm:ss :  2021-04-13 11:18:31
Time with MicroSeconds:  2021-04-13 11:18:31.635281
Time with NanoSeconds:  2021-04-13 11:18:31.635281900
ShortNum Month :  2021-4-13
LongMonth :  2021-April-13
ShortMonth :  2021-Apr-13
ShortYear :  21-Apr-13
LongWeekDay :  2021-04-13 11:18:31 Tuesday
ShortWeek Day :  2021-04-13 Tue
ShortDay :  Tue 2021-04-13
Short Hour Minute Second:  2021-04-13 11:18:31
Short Hour Minute Second:  2021-04-13 11:18:31 AM
Short Hour Minute Second:  2021-04-13 11:18:31 am
PS G:\Desktop\Go Lang\lab>
```

10. **GO program with example of Array Reverse Sort Functions for integer and strings.**

```
package main //package declaration

import (
"fmt"  //format package

"sort" //package used for sorting
)

func main() { //main function

fmt.Println("integer reverse sort")  //printing statement

num := []int{50, 40, 60, 9, 80}  //array 'num' that stores the array of integers which
//needs to be sorted

sort.Sort(sort.Reverse(sort.IntSlice(num))) //Sort() method can be used to sort slices
//of strings, integers and floating point numbers.

//sort.Reverse is used to reverse sort of a slice of integers

//int type slice is sorted by using IntSlice() method.

fmt.Println(num)   //prints the sorted list of integers

fmt.Println()  //printing statement to move into next line

fmt.Println("string reverse sort")  //printing statement

text := []string{"Japan", "UK", "Germany", "Australia", "Pakistan"} //array 'text' that
//stores the array of strings which needs to be sorted
```

```
sort.Sort(sort.Reverse(sort.StringSlice(text)))

//string type slice is sorted by using StringSLice() method.

fmt.Println(text) //prints the sorted list of string
}
```

OUPUT: