

**Lab Manual**  
**on**  
**Go Programming Lab**  
**20-CS-PC-412**  
**(IV- B. Tech. – I– Semester)**  
**Submitted to**

**Department of Computer Science & Engineering (Data Science)**

**By**

Ms.K.Yamuna

(Asst. Professor, Dept. of CSE(DS))



**CMR INSTITUTE OF TECHNOLOGY**

Kandlakoya(V), Medchal Road, Hyderabad – 501 401

Ph. No. 08418-222042, 22106 Fax No. 08418-222106

**(2023-24)**

**Vision:** To create world class technocrats for societal needs.

**Mission:** Achieve global quality technical education by assessing learning environment through

- Innovative Research & Development
- Eco-system for better Industry institute interaction
- Capacity building among stakeholders

**Quality Policy:** Strive for global professional excellence in pursuit of key-stakeholders.

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DS)

**Vision:** Develop competent software professionals, researchers and entrepreneurs to serve global society.

**Mission:** The department of **Computer Science and Engineering (DS)** is committed to

- create technocrats with proficiency in design and code for software development
- adapt contemporary technologies by lifelong learning and face challenges in IT and ITES sectors
- quench the thirst of knowledge in higher education, employment, R&D and entrepreneurship

**Program educational objectives (PEOs):** Engineering Graduates will

1. Pursue successful professional career in IT and IT-enabled sectors.
2. Pursue lifelong learning skills to solve complex problems through multidisciplinary-research.
3. Exhibits professionalism, ethics and inter-personal skills to develop leadership qualities.

**Programme Outcomes (POs):** Engineering Graduates will be able to

1. Apply mathematics, science, engineering fundamentals to solve complex engineering problems.
2. Identify, formulate and analyze complex engineering problems to reach substantiated conclusions.
3. Design and develop a component/system/process to solve complex societal engineering problems.
4. Design and conduct experiments to analyze, interpret and synthesize data for valid conclusions.
5. Create, select and apply modern tools, skills, resources to solve complex engineering problems.
6. Apply contextual engineering knowledge to solve societal issues.
7. Adapt modern engineering practices with environmental safety and sustainable development.
8. Apply professional code of ethics, responsibilities and norms in engineering practices.
9. Compete as an individual and/or as a leader in collaborative cross cultural teams.
10. Communicate effectively through technical reports, designs, documentations and presentations.
11. Endorse cognitive management skills to prepare project report using modern tools and finance.
12. Engage in independent and life-long learning in the broad context of technological changes.

**Programme Specific Outcomes (PSOs):** Engineering Graduates will be able to

1. Design and develop Computer-Based-Systems using Algorithms, Networks, Security, Gaming, Full Stack, DevOps, IoT, Cloud, Data Science and AI & ML.
2. Apply techniques of AI & ML to solve real world problems.

**GO Programming LAB**

**III-B.Tech.-II-Sem.**  
**Subject Code: 20-CS-PCC-412**

**L T P C**  
**- - 2 1**

**Course Outcomes:**

<b>COs</b>	<b>Upon completion of the course, the students will be able to:</b>	<b>PO4</b>	<b>PO5</b>	<b>PSO2</b>
<b>CO1</b>	Illustrate the implementation procedures for the machine learning algorithms	3	3	3
<b>CO2</b>	Demonstrate the ID3 Classification algorithms	3	3	3
<b>CO3</b>	Analyze k-Means clustering on different datasets	3	3	3
<b>CO4</b>	Apply predictive algorithms on live data	3	3	3
<b>CO5</b>	Identify the regression algorithms to solve real world problems	3	3	3

**List of Experiments:**

1. Write a Go Program to find LCM and GCD of given two numbers.
2. Write a Go Program to print pyramid of numbers.
3. Write a program to use struct that is imported from another package.
4. Write a Go Program to calculate standard deviation in Math package.
5. Write a Program in Go language to print Floyd's Triangle.
6. Write a Go Program to take user input and addition of two strings.
7. Write a Go Program to check whether a string is Palindrome or not.
8. Write a Go Program to Build a contact form.
9. Write a Go Program to calculate average using arrays.

10. Write a Go program to delete duplicate element in a given array.
11. Write a Go Program with example of Array Reverse Sort Functions for integer and strings.
12. Write a program comprising of Contains, Contains Any, Count and Equal Fold string functions.
13. Write a Go Program for CRUD using MYSQL from scratch.
14. Write a Go Program to create multiple go routines and implement how the go routines scheduler behaves with three logical processors for CRUD using MYSQL from scratch.

**References: GO Programming Lab Manual, Department of CSE ,CMRIT,Hyd.**

**Micro-Projects:** Student must submit a report on one of the following Micro–Projects before commencement of second internal examination.

1. Build a database using Go Programming.
2. Create a calculator in Go Programming.
3. Create a countdown using Go Programming.
4. Create a Tic Tac Toe using Go Programming.
5. Convert a text file to PDF using Go Programming.
6. Build a simple website using Go Programming.
7. Build a book management system using Go Programming
8. Build a restaurant management system using Go Programming.
9. Build a office management system using Go Programming.
10. Build a simple server in Go Programming.

---

**INDEX**

<b>Week No.</b>	<b>Name of the Experiment</b>	<b>Page</b>
1	Write a Go Program to find LCM and GCD of given two numbers.	7-10
2	Write a Go Program to print pyramid of numbers.	11-12
3	Write a program to use struct that is imported from another package.	13-15
4	Write a Go Program to calculate standard deviation in Math package.	16-17
5	Write a Program in Go language to print Floyd's Triangle.	18-20
6	Write a Go Program to take user input and addition of two strings.	21-22
7	Write a Go Program to check whether a string is Palindrome or not.	23-24
8	Write a Go Program to Build a contact form.	25-28
9	Write a Go Program to calculate average using arrays.	29-31
10	Write a Go program to delete duplicate element in a given array.	32-33
11	Write a Go Program with example of Array Reverse Sort Functions for integer and strings.	34-35
12	Write a program comprising of Contains, Contains Any, Count and Equal Fold string functions.	36-37
13	Write a Go Program for CRUD using MYSQL from scratch.	38-45
14	Write a Go Program to create multiple go routines and implement how the go routines scheduler behaves with three logical processors for CRUD using MYSQL from scratch.	46-49

---

**EXPERIMENT-1**

**Write a Go Program to find LCM and GCD of given two numbers.**

```
package main
import "fmt"

/* Function without return declaration*/
func lcm(temp1 int,temp2 int) {
    var lcmnum int =1
    if(temp1>temp2) {
        lcmnum=temp1
    }else{
        lcmnum=temp2
    }
    /* Use of For Loop as a While Loop*/
    for {
        if(lcmnum%temp1==0 && lcmnum%temp2==0) { // And operator
            /* Print Statement with multiple variables */
            fmt.Printf("LCM of %d and %d is %d",temp1,temp2,lcmnum)
            break
        }
        lcmnum++
    }
    return // Return without any value
}

func gcd(temp1 int,temp2 int){
    var gcdnum int
    /* Use of And operator in For Loop */
    for i := 1; i <=temp1 && i <=temp2 ; i++ {
        if(temp1%i==0 && temp2%i==0) {
            gcdnum=i
        }
    }
    fmt.Printf("GCD of %d and %d is %d",temp1,temp2,gcdnum)
    return
}

func main() {
    var n1,n2,action int
    fmt.Println("Enter two positive integers : ")
    fmt.Scanln(&n1)
    fmt.Scanln(&n2)
    fmt.Println("Enter 1 for LCM and 2 for GCD")
    fmt.Scanln(&action)
```

---

```
/* Use of Switch Case in Golang */
switch action {
    case 1:
        lcm(n1,n2)
    case 2:
        gcd(n1,n2)
}
```

Output:

Enter two positive integers :

3

6

Enter 1 for LCM and 2 for GCD

1

LCM of 3 and 6 is 6



**Viva Questions & Answers:****1. What is import “fmt”**

“fmt” is a Go package that is used to format basic strings, values, inputs, and outputs. It can also be used to print and write from the terminal.

**2. If GCD of two number is 8 and LCM is 144, then what is the second number if first number is 72?**

- a) 24
- b) 2
- c) 3
- d) 16

**Answer: d**

**Explanation:** As  $A * B = \text{GCD}(A, B) * \text{LCM}(A, B)$ . So  $B = (144 * 8) / 72 = 16$ .

**3. Which of the following is also known as GCD?**

- a) Highest Common Divisor
- b) Highest Common Multiple
- c) Highest Common Measure
- d) Lowest Common Multiple

**Answer: a**

**Explanation:** GCM (Greatest Common Measure), GCF (Greatest Common Factor), HCF (Highest Common Factor) and HCF (Highest Common Divisor) are also known as GCD.

**4. Which of the following is coprime number?**

- a) 54 and 24
- b) 4 and 8
- c) 6 and 12
- d) 9 and 28

**Answer: d**

**Explanation:** Coprime numbers have GCD 1. So 9 and 28 are coprime numbers. While 54 and 24 have GCD 6, 4 and 8 have GCD 4, 6 and 12 have GCD 6.

**5. In terms of Venn Diagram, which of the following expression gives GCD (Given  $A \cap B \neq \emptyset$ )?**

- a) Multiplication of  $A \cup B$  terms
- b) Multiplication of  $A \cap B$  terms
- c) Multiplication of  $A * B$  terms
- d) Multiplication of  $A - B$  terms

**Answer: b**

**Explanation:** In terms of Venn Diagram, the GCD is given by the intersection of two sets. So  $A \cap B$  gives the GCD. While  $A \cup B$  gives the LCM.

---

**EXPERIMENT-2****Write a GO Program to print Pyramid of numbers.**

```
package main
import "fmt"
func main() {
    var rows,k,temp,temp1 int
    fmt.Print("Enter number of rows :")
    fmt.Scan(&rows)

    for i := 1; i <= rows; i++ {

        for j := 1; j <= rows-i; j++ {
            fmt.Print(" ")
            temp++
        }
        for{
            if( temp <= rows-1){
                fmt.Printf(" %d",i+k)
                temp++
            }else{
                temp1++
                fmt.Printf(" %d", (i+k-2*temp1))
            }
            k++

            if(k == 2*i-1) {
                break
            }

        }
        temp = 0
        temp1 = 0
        k = 0
        fmt.Println("")
    }

}
```

**Output:**

Enter number of rows : 5

```
  1
 2 3 2
3 4 5 4 3
4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
```

**Viva Questions & Answers:****1. What is Go programming language?**

**GO is an open source programming language developed at Google. It is also known as Golang. This language is designed primarily for system programming.**

**2. Who is known as the father of Go programming language?**

**Go programming language is designed by Robert Griesemer, Rob Pike, and Ken Thompson. It is developed at Google Inc. in 2009.**

**3. What are packages in Go program?**

**Go programs are made up of packages. The program starts running in package main. This program is using the packages with import paths "fmt" and "math/rand".**

**4. What is GOPATH environment variable in go programming?**

**The GOPATH environment variable specifies the location of the workspace. You must have to set this environment variable while developing Go code.**

**5. What is workspace in Go?**

**A workspace contains Go code. A workspace is a directory hierarchy with three directories at its root.**

- **"src" directory contains GO source files organized into packages.**
- **"pkg" directory contains package objects.**
- **"bin" directory contains executable commands**

---

**EXPERIMENT-3**

**Write a program to use struct that is imported from another package.**

```
package main

import (
    parent "family/father"
    child  "family/father/son"

    "fmt"
)

func main() {
    f := new(parent.Father)
    fmt.Println(f.Data("Mr. Jeremy Maclin"))

    c := new(child.Son)
    fmt.Println(c.Data("Riley Maclin"))
}

package father

import "fmt"

func init() {
    fmt.Println("Father package initialized")
}

type Father struct {
    Name string
}

func (f Father) Data(name string) string {
    f.Name = "Father : " + name
    return f.Name
}
```

---

```
package son

import "fmt"

func init() {

    fmt.Println("Son package initialized")

}

type Son struct {

    Name string

}

func (s Son) Data(name string) string {

    s.Name = "Son : " + name

    return s.Name

}
```

**Output:**

```
Father package initialized
Son package initialized
Father : Mr. Jeremy Maclin
Son : Riley Maclin
```

**Viva Questions & Answers:****1. What are packages in Go program?**

**Ans:** Go programs are made up of packages. The program starts running in package main. This program is using the packages with import paths "fmt" and "math/rand".

**2. Is Go a case sensitive language?**

**Ans:** Yes! Go is a case sensitive programming language.

**3. What is GOPATH environment variable in go programming?**

**Ans:** The GOPATH environment variable specifies the location of the workspace. You must have to set this environment variable while developing Go code.

**4. Benefits of Go programming language?**

**Ans:** Advantages/ Benefits of Go programming language:

- Go is fast and compiles very quickly.
- It supports concurrency at the language level.
- It has Garbage collection.
- It supports various safety features and CSP-style concurrent programming features.
- Strings and Maps are built into the language.
- Functions are first class objects in this language.

**5. How would you print type of variable in Go?**

**Ans:** You have to use the following code to print the type of a variable:

1. `var a, b, c = 3, 4, "foo"`
2. `fmt.Printf("a is of type %T\n", a)`

---

**EXPERIMENT-4****Write a GO Program to calculate Standard deviation in math package**

```
package main
import (
    "fmt"
    "math"
)
func main() {
    var num[10] float64
    var sum, mean, sd float64
    fmt.Println("***** Enter 10 elements *****")
    for i := 1; i <= 10; i++ {
        fmt.Printf("Enter %d element : ", i)
        fmt.Scan(&num[i-1])
        sum += num[i-1]
    }
    mean = sum/10;

    for j := 0; j < 10; j++ {
        // The use of Pow math function func Pow(x, y float64)
        float64
        sd += math.Pow(num[j] - mean, 2)
    }
    // The use of Sqrt math function func Sqrt(x float64) float64
    sd = math.Sqrt(sd/10)

    fmt.Println("The Standard Deviation is : ", sd)
}
```

**Output:**

```
***** Enter 10 elements *****
Enter 1 element : 3
Enter 2 element : 5
Enter 3 element : 9
Enter 4 element : 1
Enter 5 element : 8
Enter 6 element : 6
Enter 7 element : 58
Enter 8 element : 9
Enter 9 element : 4
Enter 10 element : 10
```

**The Standard Deviation is : 15.8117045254457**



---

## Viva Questions & Answers:

### 1. What is Go?

**Ans:** Go is also known as GoLang, it is a general-purpose programming language designed at Google and developed by Robert Griesemer, Ken Thomson and Rob Pike. It is a statically typed programming language used for building fast and reliable applications.

### 2. What is the static type declaration of a variable in Golang?

**Ans:** Static type variable declaration gives confirmation to the compiler that there is one variable existing with the given kind and name so the compiler continues for an additional compilation without requiring total insight concerning the variable. A variable declaration holds its importance at the moment of compilation only, the compiler requires actual variable declaration at the moment of connecting to the program.

### 3. What is the dynamic variable declaration in Golang?

**Ans:** A dynamic kind variable declaration needs the compiler to explain the kind of the variable based on the amount transferred to it. The compiler doesn't need a variable to categorise statically as a mandatory condition.

### 4. Explain pointers in Go?

**Ans:** Pointers are variables that hold the address of any variable. Pointers in Golang are likewise called special variables. There are two operators in pointers they are

- \* operator which is also called a dereferencing operator used to access the value in the address
- & operator which is also called as address operator this is utilized to return the address of the variable

### 5. What is a constant variable in Go?

**Ans:** As the name suggests constant means fixed and the meaning doesn't change in a programming language. Once the value of a constant variable is defined then it should be the same throughout the program, we cannot change the value of a variable in between the program

---

**EXPERIMENT-5**

**Write a program in GO language to print Floyd's 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 := 1; i <= rows; i++ {

        for k := 1; k <= i; k++ {

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

**Output:**

Enter number of rows : 5

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

**Viva Questions & Answers:**

1. How can you format a string without printing?

**Ans:** You should the following command to format a string without printing:

```
return fmt.Sprintf ("at %v, %s" , e.When , e.What )
```

2. Does Go programming language support type inheritance?

**Ans:** Go programming language doesn't provide support for type inheritance.

3. Does Go programming language support operator overloading?

**Ans:** Go programming language doesn't provide support for operator overloading.

4. Does Go support method overloading?

**Ans:** Go programming language doesn't provide support for method overloading.

5. What will be the output of the following code?

```
1. package main
2. import "fmt"
3. const (
4.     i = 7
5.     j
6.     k
7. )
8. func main() {
9.     fmt.Println(i, j, k)
10. }
```

**Ans:** 777

**Viva Questions & Answers:****1. How to write multiple strings in Go programming?**

**Ans:** To write multiple strings in Go, you should use a raw string literal, where the string is delimited by back quotes.

**For example:**

```
'line 1  
line 2  
line 3 '
```

**2. What is the usage of break statement in Go programming language?**

**Ans:** The break statement is used to terminate the for loop or switch statement and transfer execution to the statement immediately following the for loop or switch.

**3. What is the usage of continue statement in Go programming language?**

**Ans:** The continue statement facilitates the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

**4. What is the usage of goto statement in Go programming language?**

**Ans:** The goto statement is used to transfer control to the labeled statement.

**5. Explain the syntax for 'for' loop.**

**Ans:** The syntax of a for loop in Go programming language is:

```
for [condition | ( init; condition; increment ) | Range]  
{  
    statement(s);  
}
```

---

**EXPERIMENT-6**

**Write a GO Program to take user input and addition of two strings.**

```
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)
    fmt.Print(first + second)         // Addition of two
    string
}
```

**Output:**

```
Enter First String: Go
Enter Second String: Programming
GoProgramming
```

**Viva Questions & Answers:****1. List the operators in Golang?****Ans:**

- Arithmetic operators
- Bitwise operators
- Relational operators
- Logical operators
- Assignment operators
- Misc operators

**2. List data types on Golang?****Ans:** There are 4 data types in the Go language

- Basic type numbers, strings, and booleans
- Aggregate type structures and arrays
- Reference type slices, pointers, maps, channels, and functions
- Interface type

**3. What is the scope of a variable?**

**Ans:** The scope of a variable means the part of a program where the particular variable can be accessed. In the Go language, every variable is statistically scoped that means the scope of a variable is declared at compile time itself.

Scope of a variable in the Go language is categorized into two types

- Local variables these variables are either declared inside a function or a block
- Global variables these variables are declared outside the function or a block

**4. Explain Methods in Golang?**

**Ans:** There is only one difference between Go methods and Go functions that is the methods of Golang contain receiver argument in them. The method can obtain the characteristics of the receiver with the cooperation of the receiver argument.

**syntax:**

```
func(name type) method_name(param_list)(return_type)
```

```
{
```

```
//code
```

```
}
```

**5. Is GoLang fast?**

**Ans:** Golang's concurrency model and small syntax make Golang fast programming language, Golang compilation is very fast, Go hyperlinks all the dependency libraries into a single binary file, as a result, putting off the dependence on servers.

---

**EXPERIMENT-7**

**Write a GO Program to check whether a string is Palindrome or not.**

```
package main
import "fmt"

// start the function main ()
func main() {
    fmt.Println("Golang program to check for palindrome")

    // declare the variables
    var number, rem, temporary int
    var reverse int = 0

    // initialize the number variable
    number = 45454
    temporary = number

    // For Loop used
    for {
        rem = number%10
        reverse = reverse*10 + rem
        number /= 10
        if (number==0) {
            break // Break Statement used to exit from loop
        }
    }
    if (temporary==reverse) {
        fmt.Printf("Number %d is a Palindrome", temporary)
    } else {
        fmt.Printf("Number %d is not a Palindrome", temporary)
    }
    // print the result using fmt.Printf () function
}
```

**Output:**

Enter any positive integer : 5454  
5454 is not a Palindrome

---

**Viva Questions & Answers:****1. Explain what is string types?**

**Ans:** A string type represents the set of string values, and string values are sequence of bytes. Strings once created is not possible to change.

**2. Explain how arrays in GO works differently then C ?**

**Ans:** In GO Array works differently than it works in C

- Arrays are values, assigning one array to another copies all the elements
- If you pass an array to a function, it will receive a copy of the array, not a pointer to it
- The size of an array is part of its type. The types [10] int and [20] int are distinct

**3. Explain what Type assertion is used for and how it does it?**

**Ans:** Type conversion is used to convert dissimilar types in GO. A type assertion takes an interface value and retrieve from it a value of the specified explicit type.

**4. What do you know about modular programming?**

**Ans:** Modular programming is a way to divide the program in to sub programs (modules / function) to achieve maximum efficiency.

More generic functions definition facilitates you to re-use the functions, such as built-in library functions.

**5. What are the function closures?**

**Ans:** Functions closure are anonymous functions and can be used in dynamic programming



---

**EXPERIMENT-8**
**Write a GO Program to Build a Contact form**

```

package main

import (
    "html/template"
    "net/http"
)

type ContactDetails struct {
    Email    string
    Subject  string
    Message  string
}

func main() {
    tmpl := template.Must(template.ParseFiles("forms.html"))

    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request)
    {
        if r.Method != http.MethodPost {
            tmpl.Execute(w, nil)
            return
        }

        details := ContactDetails{
            Email:    r.FormValue("email"),
            Subject: r.FormValue("subject"),
            Message: r.FormValue("message"),
        }

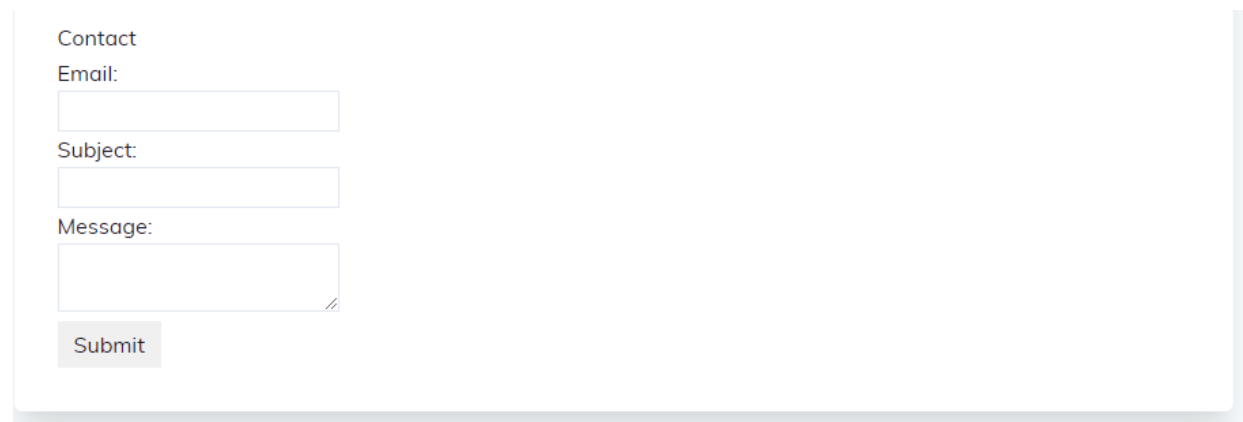
        // do something with details
        _ = details

        tmpl.Execute(w, struct{ Success bool }{true})
    })

    http.ListenAndServe(":8080", nil)
}
<!-- forms.html -->
{{if .Success}}
    <h1>Thanks for your message!</h1>
{{else}}
    <h1>Contact</h1>
    <form method="POST">
        <label>Email:</label><br />
        <input type="text" name="email"><br />
        <label>Subject:</label><br />
        <input type="text" name="subject"><br />
    </form>

```

```
<label>Message:</label><br />
<textarea name="message"></textarea><br />
<input type="submit">
</form>
{{end}}
```

**OUTPUT:**

Contact

Email:

Subject:

Message:

Submit

## Viva Questions & Answers:

### 1. How to find the type of Struct in Golang?

**Ans:** A structure or struct in Golang is a user-defined data type which is a composition of various data fields. Each data field has its own data type, which can be a built-in or another user-defined type. Struct represents any real-world entity that has some set of properties/fields. Go does not support the concept of classes, structs are the only way to create a user-defined type in this language. There are various ways by which we can identify the type of struct in Go:

#### Method 1: *Using reflect package*

You can use the reflect package to find the given type of a struct. Reflection package allows determining the variables' type at runtime.

Syntax:

```
func typeofstruct(x interface{}){  
    fmt.Println(reflect.TypeOf(x))  
}
```

or

```
func typeofstruct(x interface{}){  
    fmt.Println(reflect.ValueOf(x).Kind())  
}
```

### 2. what is http.HandleFunc

**Ans:** This is an interface. Anything that implements this interface is a handler. Implementing this interface means any function which has signature matching to `ServeHTTP(ResponseWriter, *Request)`. This is what official documentation says.

```
type Handler interface {  
    ServeHTTP(ResponseWriter, *Request)  
}
```

The pointer to Request above is an struct which holds data from the client. ResponseWriter is an interface which allows us to respond to the request elegantly.

---

```
func ServeHTTP(w http.ResponseWriter, req *http.Request) {  
    io.WriteString(w, "I am writing to the response")  
}
```

**http.ResponseWriter** has a method signature of **Write([]byte) (int, error)** which means any type that implements **io.Writer** can also write to the **w**; **io.WriteString** being one of them

### 3. What are the decision-making statements in Golang?

**Ans:** There are 4 decision-making statements in Golang they are

- **if statement:** used to decide whether certain statements will be executed or not
- **if-else statement:** if a certain condition is true only then it will execute a block if it is not true then it won't execute the block of code
- **Nested-if statement:** nested if means if condition inside another
- **if condition**
- **if-else-if ladder:** Here, a user can choose among various options. The if statements are executed of top-down. As early as one of the states controlling the if is correct, the statement compared with that if is executed, and the remaining ladder is bypassed. If none of the conditions is correct, then the last else statement will be executed.

### 4. Which kind of conversion is supported by Golang?

**Ans:** Go is very particular about explicit typing. There is no automated type conversion. Explicit type conversion is needed to designate a variable of one type to another.

### 5. What is CGo in Golang?

**Ans:** CGo allows Go packages to invite C code. Given a Go source file that is written with some unique features, cGo yields Go and C files that can be merged into a unique Go package. That is because C means a "pseudo-package", a unique name defined by cGo as a reference to C's namespace

---

**EXPERIMENT-9**

**Write a GO Program to calculate average using arrays**

```
package main

import "fmt"

func main() {
    var num[100] int
    var temp, sum, avg int
    fmt.Print("Enter number of elements: ")
    fmt.Scanln(&temp)
    for i := 0; i < temp; i++ {
        fmt.Print("Enter the number : ")
        fmt.Scanln(&num[i])
        sum += num[i]
    }

    avg = sum/temp
    fmt.Printf("The Average of entered %d number(s) is
%d", temp, avg)
}
```

**OUTPUT :**

```
Enter number of elements: 5
Enter the number : 2
Enter the number : 65
Enter the number : 18
Enter the number : 59
Enter the number : 54
The Average of entered 5 number(s) is 39
```

## Viva Questions & Answers:

### 1. What is Golang arrays.

**Ans:** An array is a numbered sequence of an element with a specified length in Golang (Go Language). For an example, `arr [n]T` : means `arr` is an array of `n` element of type `T`.

Consider the below expression,

```
var x[5]int
```

It declares an array `x` with `5` elements of `int` type.

### 2. What is shorthand array declaration.

**Ans:** `package main`

```
import "fmt"
```

```
func main() {  
    arr := [5]int{1, 2, 3, 4, 5}  
  
    fmt.Println("Array elements:")  
    for i := 0; i <= 4; i++ {  
        fmt.Printf("%d ", arr[i])  
    }  
}
```

**Output:**

```
Array elements:  
1 2 3 4 5
```

In the `main()` function, we created and initialized an integer array `arr` of 5 elements using the shorthand declaration method. After that, we printed the array elements using the `for` loop on the console screen.

### 3. How to create an array without specifying its size.

**Ans:** `package main`

```
import "fmt"
```

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

```
    fmt.Println("Array elements:")
    for i := 0; i <= 7; i++ {
        fmt.Printf("%d ", arr[i])
    }
}
```

Output:

Array elements:  
1 2 3 4 5 6 7 8

In the *main()* function, we created and initialized an integer array *arr* using *"..."* without specifying the size of the array. After that, we printed the array elements using the *"for"* loop on the console screen.

#### 4. What are Lvalue in Golang?

Ans: Lvalue

- Refers to a memory location
- Represents a variable identifier
- Mutable
- May appear on the left or right side of the `=` operator

For example: In the statement `x=20`, `x` is an lvalue and `20` is an rvalue.

#### 5. What is RLvalue in Golang?

Ans: Rvalue

- Represents a data value stored in memory
- Represents a constant value
- Always appears on the `=` operator's right side.

For example, The statement `10 = 20` is invalid because there is an rvalue (10) left of the `=` operator.

---

**EXPERIMENT-10****Write a GO Program to delete duplicate element in a given array**

```
package main
import "fmt"

// function to remove duplicate values
func removeDuplicates(s []string) []string {
    bucket := make(map[string]bool)
    var result []string
    for _, str := range s {
        if _, ok := bucket[str]; !ok {
            bucket[str] = true
            result = append(result, str)
        }
    }
    return result
}

func main() {

    // creating an array of strings
    array := []string{"abc", "cde", "efg", "efg", "abc", "cde"}
    fmt.Println("The given array of string is:", array)
    fmt.Println()

    // calling the function
    result := removeDuplicates(array)
    fmt.Println("The array obtained after removing the duplicate entries is:", result)
}
```

**OUTPUT :**

The given array of string is: [abc cde efg efg abc cde]

The array obtained after removing the duplicate entries is: [abc cde efg]



---

### Viva Questions & Answers:

1. Is it possible to declare variables of different types in a single line of code in Golang?

**Ans:** Yes, this can be achieved by writing as shown below:

```
var a,b,c= 9, 7.1, "interviewbit"
```

Here, we are assigning values of a type Integer number, Floating-Point number and string to the three variables in a single line of code

2. Is the usage of Global Variables in programs implementing goroutines recommended?

**Ans:** Using global variables in goroutines is not recommended because it can be accessed and modified by multiple goroutines concurrently. This can lead to unexpected and arbitrary results.

3. What are the uses of an empty struct?

**Ans:** Empty struct is used when we want to save memories. This is because they do not consume any memory for the values. The syntax for an empty struct is:

```
a := struct{}{}
```

The size of empty struct would return 0 when using `println(unsafe.Sizeof(a))`

4. What do you understand by byte and rune data types? How are they represented?

**Ans:** byte and rune are two integer types that are aliases for uint8 and int32 types respectively.

The byte represents ASCII characters whereas the rune represents a single Unicode character which is UTF-8 encoded by default.

- The characters or rune literals can be represented by enclosing in single quotes like 'a','b','\n'.
- Rune is also called a Code point and can also be a numeric value. For example, 0x61 in hexadecimal corresponds to the rune literal a

5. What is default value of a global variable in Go?

**Ans:** The default value of a local variable is as its corresponding 0 value.

---

**EXPERIMENT-11**

**Write a GO Program with an example of Array reverse sort Functions for integers and strings.**

```
package main

import (
    "fmt"
    "sort"
)

func main() {

    fmt.Println("Integer Reverse Sort")
    num := []int{50, 90, 30, 10, 50}
    sort.Sort(sort.Reverse(sort.IntSlice(num)))
    fmt.Println(num)

    fmt.Println()

    fmt.Println("String Reverse Sort")
    text := []string{"Japan", "UK", "Germany", "Australia", "Pakistan"}
    sort.Sort(sort.Reverse(sort.StringSlice(text)))
    fmt.Println(text)

}
```

**Output :**

Integer Reverse Sort  
[90 50 50 30 10]

String Reverse Sort  
[UK Pakistan Japan Germany Australia]

---

**Viva Questions & Answers:****1. Why is Golang reliable?**

**Ans:** Golang is reliable because it's type-safe, making it harder to crash the program. Additionally, it's impossible to misinterpret any type on Golang.

**2. Does Golang take 'cases' into account?**

**Ans:** Go is one of the only languages that truly incorporates the case of identifiers into its fundamental syntax and is case-sensitive. An object's type, method, etc., is "exported" (visible to other packages) if its name begins with an uppercase letter but not if it begins with a lowercase letter.

**3. What are variadic functions in Go?**

**Ans:** A variadic function accepts various numbers of arguments. Golang allows a variable number of parameters of the same type as those listed in the function signature.

**4. Differentiate between const and read-only keywords.**

**Ans:**

**const:**

- These variables are evaluated at compile time
- They are only for value types

**read-only:**

- These variables are evaluated at runtime
- They can hold reference type variables

**5. What is heap memory?**

**Ans:** We store objects and variables created dynamically in heap memory. When we no longer need an item, we frequently erase the memory it used up on the heap.

**EXPERIMENT-12**

**Write a program comprising of Contains , Contains Any , Count and Equal Fold String functions.**

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

func main() {
    fmt.Println(strings.ContainsAny("Germany", "G"))
    fmt.Println(strings.ContainsAny("Germany", "g"))

    fmt.Println(strings.Contains("Germany", "Ger"))
    fmt.Println(strings.Contains("Germany", "ger"))
    fmt.Println(strings.Contains("Germany", "er"))

    fmt.Println(strings.Count("cheese", "e"))

    fmt.Println(strings.EqualFold("Cat", "cAt"))
    fmt.Println(strings.EqualFold("India", "Indiana"))
}
```

#### Output:

```
true
false
true
false
true
3
true
false
```

#### Viva Questions & Answers:

1. Can you change a specific character in a string?

**Ans:** No. Strings are immutable (read-only) data types and you cannot change them. If we try to change a specific character in a string, we'll get a runtime error.

2. How can you concatenate string values? What happens when concatenating strings?

**Ans:** To concatenate strings, we can use the addition operator (+). Note that each time we concatenate to a string value, Go creates a new string. That's because strings are immutable in Go, making this inefficient.

There is an efficient and recommended way to concatenate string values and that is to use the `strings.Builder` type, which was introduced in Go 1.10.

3. Explain array and slice types and the differences between them.

**Ans:** Golang has two data structures to handle lists of records: arrays and slices.

An array is a composite, indexable type that stores a collection of elements.

An array has a fixed length. We specify how many items are in the array when we declare it. This is in contrast to a slice that has a dynamic length (it can shrink or grow at runtime).

The array length is part of its type.

Every element in an array or slice must be of the same type.

Slices are a key data type in Golang and are everywhere.

4. Give an example of an array and slice declaration.

**Ans:** Here's an example of declaring and initializing an array of type `[4] string` using the short declaration syntax.

```
friends := [4]string{"Dan", "Diana", "Paul", "John"}
```

5. Explain the Object-Oriented Architecture of Golang.

**Ans:** Unlike traditional Object-Oriented Programming, Golang does not have a class-object architecture. Rather structs and methods hold complex data structures and behavior.

*A struct is nothing more than a schema containing a blueprint of data that a structure will hold. Structs are useful to represent concepts from the real world like cars, people, or books.*

---

**EXPERIMENT-13**
**Write a GO Program for CRUD using MYSQL from scratch**

```

DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL,
  `city` varchar(30) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

package main

import (
    "database/sql"
    "log"
    "net/http"
    "text/template"

    _ "github.com/go-sql-driver/mysql"
)

type Employee struct {
    Id      int
    Name    string
    City    string
}

func dbConn() (db *sql.DB) {
    dbDriver := "mysql"
    dbUser := "root"
    dbPass := "root"
    dbName := "goblog"
    db, err := sql.Open(dbDriver, dbUser+": "+dbPass+"@"+dbName)
    if err != nil {
        panic(err.Error())
    }
    return db
}

var tmpl = template.Must(template.ParseGlob("form/*"))

func Index(w http.ResponseWriter, r *http.Request) {

```

---

```

    db := dbConn()
    selDB, err := db.Query("SELECT * FROM Employee ORDER BY id DESC")
    if err != nil {
        panic(err.Error())
    }
    emp := Employee{}
    res := []Employee{}
    for selDB.Next() {
        var id int
        var name, city string
        err = selDB.Scan(&id, &name, &city)
        if err != nil {
            panic(err.Error())
        }
        emp.Id = id
        emp.Name = name
        emp.City = city
        res = append(res, emp)
    }
    tpl.ExecuteTemplate(w, "Index", res)
    defer db.Close()
}

func Show(w http.ResponseWriter, r *http.Request) {
    db := dbConn()
    nId := r.URL.Query().Get("id")
    selDB, err := db.Query("SELECT * FROM Employee WHERE id=?", nId)
    if err != nil {
        panic(err.Error())
    }
    emp := Employee{}
    for selDB.Next() {
        var id int
        var name, city string
        err = selDB.Scan(&id, &name, &city)
        if err != nil {
            panic(err.Error())
        }
        emp.Id = id
        emp.Name = name
        emp.City = city
    }
    tpl.ExecuteTemplate(w, "Show", emp)
    defer db.Close()
}

func New(w http.ResponseWriter, r *http.Request) {
    tpl.ExecuteTemplate(w, "New", nil)
}

```

---

```

func Edit(w http.ResponseWriter, r *http.Request) {
    db := dbConn()
    nId := r.URL.Query().Get("id")
    selDB, err := db.Query("SELECT * FROM Employee WHERE id=?", nId)
    if err != nil {
        panic(err.Error())
    }
    emp := Employee{}
    for selDB.Next() {
        var id int
        var name, city string
        err = selDB.Scan(&id, &name, &city)
        if err != nil {
            panic(err.Error())
        }
        emp.Id = id
        emp.Name = name
        emp.City = city
    }
    tmpl.ExecuteTemplate(w, "Edit", emp)
    defer db.Close()
}

func Insert(w http.ResponseWriter, r *http.Request) {
    db := dbConn()
    if r.Method == "POST" {
        name := r.FormValue("name")
        city := r.FormValue("city")
        insForm, err := db.Prepare("INSERT INTO Employee(name, city)
VALUES(?,?)")
        if err != nil {
            panic(err.Error())
        }
        insForm.Exec(name, city)
        log.Println("INSERT: Name: " + name + " | City: " + city)
    }
    defer db.Close()
    http.Redirect(w, r, "/", 301)
}

func Update(w http.ResponseWriter, r *http.Request) {
    db := dbConn()
    if r.Method == "POST" {
        name := r.FormValue("name")
        city := r.FormValue("city")
        id := r.FormValue("uid")
        insForm, err := db.Prepare("UPDATE Employee SET name=?,
city=? WHERE id=?")

```



---

```

        if err != nil {
            panic(err.Error())
        }
        insForm.Exec(name, city, id)
        log.Println("UPDATE: Name: " + name + " | City: " + city)
    }
    defer db.Close()
    http.Redirect(w, r, "/", 301)
}

func Delete(w http.ResponseWriter, r *http.Request) {
    db := dbConn()
    emp := r.URL.Query().Get("id")
    delForm, err := db.Prepare("DELETE FROM Employee WHERE id=?")
    if err != nil {
        panic(err.Error())
    }
    delForm.Exec(emp)
    log.Println("DELETE")
    defer db.Close()
    http.Redirect(w, r, "/", 301)
}

func main() {
    log.Println("Server started on: http://localhost:8080")
    http.HandleFunc("/", Index)
    http.HandleFunc("/show", Show)
    http.HandleFunc("/new", New)
    http.HandleFunc("/edit", Edit)
    http.HandleFunc("/insert", Insert)
    http.HandleFunc("/update", Update)
    http.HandleFunc("/delete", Delete)
    http.ListenAndServe(":8080", nil)
}

{{ define "Index" }}
{{ template "Header" }}
{{ template "Menu" }}
<h2> Registered </h2>
<table border="1">
    <thead>
        <tr>
            <td>ID</td>
            <td>Name</td>
            <td>City</td>
            <td>View</td>
            <td>Edit</td>
            <td>Delete</td>

```

---

```

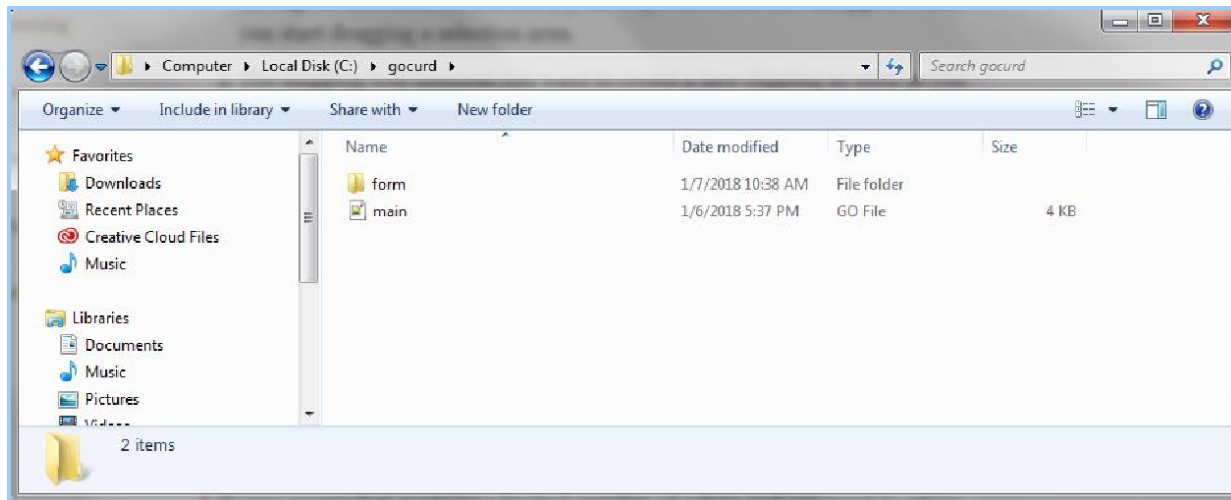
        </tr>
    </thead>
    <tbody>
    {{ range . }}
        <tr>
            <td>{{ .Id }}</td>
            <td> {{ .Name }} </td>
            <td>{{ .City }} </td>
            <td><a href="/show?id={{ .Id }}">View</a></td>
            <td><a href="/edit?id={{ .Id }}">Edit</a></td>
            <td><a href="/delete?id={{ .Id }}">Delete</a><td>
        </tr>
    {{ end }}
    </tbody>
</table>
{{ template "Footer" }}
{{ end }}
{{ define "Header" }}
<!DOCTYPE html>
<html lang="en-US">
    <head>
        <title>Golang Mysql Curd Example</title>
        <meta charset="UTF-8" />
    </head>
    <body>
        <h1>Golang Mysql Curd Example</h1>
    {{ end }}
    {{ define "Footer" }}
        </body>
</html>
{{ end }}
{{ define "Menu" }}
<a href="/">HOME</a> |
<a href="/new">NEW</a>
{{ end }}
{{ define "Show" }}
    {{ template "Header" }}
    {{ template "Menu" }}
    <h2> Register {{ .Id }} </h2>
    <p>Name: {{ .Name }}</p>
    <p>City: {{ .City }}</p><br /> <a href="/edit?id={{ .Id
}}">Edit</a></p>
    {{ template "Footer" }}
{{ end }}
{{ define "New" }}
    {{ template "Header" }}
    {{ template "Menu" }}
    <h2>New Name and City</h2>
    <form method="POST" action="insert">

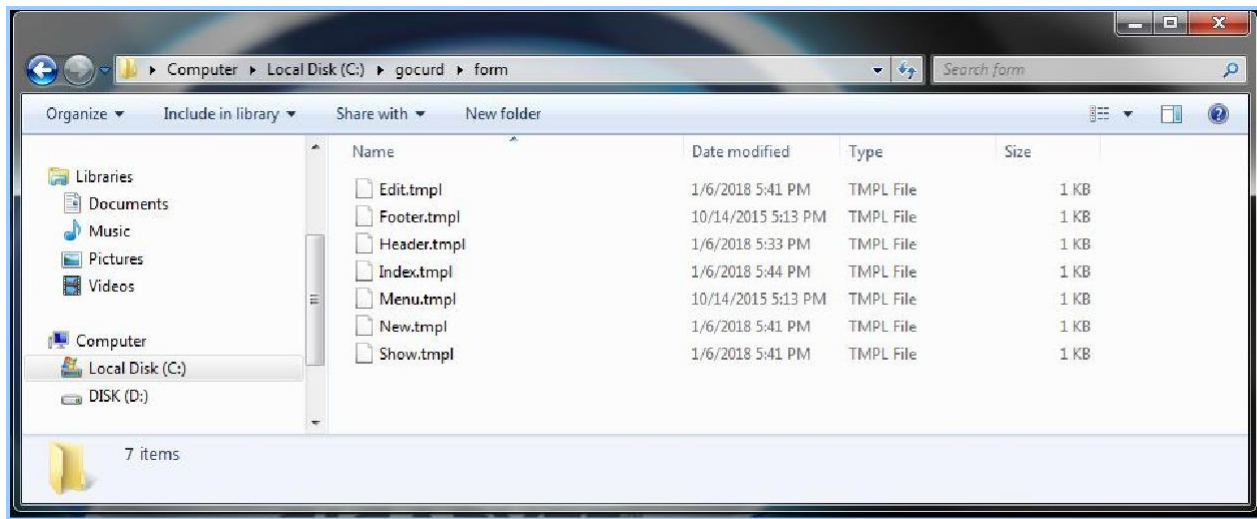
```

```

        <label> Name </label><input type="text" name="name" /><br />
        <label> City </label><input type="text" name="city" /><br />
        <input type="submit" value="Save user" />
    </form>
    {{ template "Footer" }}
{{ end }}
{{ define "Edit" }}
    {{ template "Header" }}
    {{ template "Menu" }}
    <h2>Edit Name and City</h2>
    <form method="POST" action="update">
        <input type="hidden" name="uid" value="{{ .Id }}" />
        <label> Name </label><input type="text" name="name" value="{{
.Name }}" /><br />
        <label> City </label><input type="text" name="city" value="{{
.City }}" /><br />
        <input type="submit" value="Save user" />
    </form><br />
    {{ template "Footer" }}
{{ end }}

```

**Output:**



**go run main.go**  
**<http://localhost:8080/>**

**Viva Questions & Answers:****1. What is CURD in MySQL**

**Ans:**MySQL provides a set of some basic but most essential operations that will help you to easily interact with the MySQL database and these operations are known as CRUD operations. CRUD refers to an acronym of four fundamental operators of persistent database applications CRUD operations explained: Create, read, update, and delete.

**2. How to store data in MySQL in GO lang.**

**Ans:** Install the driver. The standard library database/sql provides an interface to communicate with a SQL database.

**3. What is the default PORT of MySQL.**

**Ans:** Port 3306 is the default port for the classic MySQL protocol ( port ), which is used by the mysql client, MySQL Connectors, and utilities such as mysqldump and mysqlpump.

**4. Why it is called CURD.**

**Ans:** Curd is obtained by coagulating milk in a sequential process called curdling. It can be a final dairy product or the first stage in cheesemaking. The coagulation can be caused by adding rennet, a culture, or any edible acidic substance such as lemon juice or vinegar, and then allowing it to coagulate.

**5. How to create CURD in MySQL.**

**Ans:**

- **Single Query Execution.**
- **Multi Queries Execution.**
- **CRUD in PHP and MySQL With Prepared Statements.**
- **Select Query Execution.**
- **Update Query Using Prepared Statement.**
- **Delete Query Using Prepared Statement.**
- **Conclusion.**

---

**EXPERIMENT-14**

**Write a GO Program to create multiple goroutines and implement how the goroutines scheduler behaves with three logical processors for CRUD using MYSQL from scratch.**

```
package main

import (
    "fmt"
    "runtime"
    "sync"
)

func main() {
    // Allocate three logical processors for the scheduler to use.
    runtime.GOMAXPROCS(3)

    // processTest is used to wait for the program to finish.
    var processTest sync.WaitGroup
    // Add a count of three, one for each goroutine.
    processTest.Add(3)

    // Declaration of three anonymous function and create a
    // goroutine.
    go func() {
        defer processTest.Done()
        for i := 0; i < 30; i++ {
            for j := 51; j <= 100; j++ {
                fmt.Printf(" %d", j)
                if j == 100 {
                    fmt.Println()
                }
            }
        }
    }()
    go func() {
        defer processTest.Done()
        for j := 0; j < 10; j++ {
            for char := 'A'; char < 'A'+26; char++ {
                fmt.Printf("%c ", char)
                if char == 'Z' {
                    fmt.Println()
                }
            }
        }
    }()
}
```

```

go func() {
    defer processTest.Done()
    for i := 0; i < 30; i++ {
        for j := 0; j <= 50; j++ {
            fmt.Printf(" %d", j)
            if j == 50 {
                fmt.Println()
            }
        }
    }
}()

// Wait for the goroutines to finish.
processTest.Wait()
}

```

**Output:**

[illegible]

---

```

35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 A B C D E F G H I J K L M N O P Q R S T
U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

```



---

## Viva Questions & Answers:

1. What is a struct type? Can you change the struct definition at runtime?

**Ans:** A struct is a sequence of named elements called fields. Each field has a name and a type. We can also think of a struct as a collection of properties that are related together. They are useful for grouping data together to form records.

This blueprint is fixed at compile time. It's not allowed to change the name or the type of the fields at runtime. We can't add or remove fields from a struct at runtime.

2. Write a simple text on the console in Go.

**Ans:**

```
package main

import "fmt"

func main()
{
    fmt.Println("Hello World")
}
```

3. What are design patterns?

**Ans:** Design patterns are reusable pieces of code you can repeatedly use to solve common software problems. Your projects will produce more modular, scalable, and optimized software if you employ design patterns. Design patterns help you expand your apps and collaborate with a team. The Factory Method, Singleton, Facade, and Decorator are common design patterns examples.

4. Explain the distinction between methods and functions in Golang.

**Ans:** The main distinction between Go functions and Go methods is that Golang methods have receiver arguments. With the receiver argument's participation, the procedure can acquire the receiver's attributes.

5. Can we compare two structures in Go?

**Ans:** Yes, we can compare two structures in Go using a simple '==' operator.