National University of Technology

Computer Science Department

Semester Spring– 2025
Program: Artificial intelligence
Course: Programming for AI Lab
Course Code: CS283

Lab Report- 09

Submitted To:

Umar Aftab

Submitted By:

Muhammad Ahad Imran

F23607034

Go Programming

1. Hello, World!

```go
package main

import "fmt"



func main() {

    fmt.Println("Hello, Ahad!")

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t1.go"
Hello, Ahad!

[Done] exited with code=0 in 0.659 seconds
```

2. Variables & Constants

```go
package main



import "fmt"



func main() {

    var x int = 5

    const y = 10

    x += 1



    var name string = "GoLang Programmer" // Added string variable

    fmt.Printf("x = %d, y = %d\n", x, y)
```

```
    fmt.Printf("Name: %s\n", name) // Printing the string

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t2.go"
x = 6, y = 10
Name: GoLang Programmer

[Done] exited with code=0 in 0.644 seconds
```

3. Arithmetic Operations

```
package main



import "fmt"



func main() {

    a := 10

    b := 3

    fmt.Printf("Sum: %d\n", a+b)

    fmt.Printf("Product: %d\n", a*b)



    // Calculate remainder of 15 divided by 4

    remainder := 15 % 4

    fmt.Printf("Remainder of 15/4: %d\n", remainder)

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t3.go"
Sum: 13
Product: 30
Remainder of 15/4: 3

[Done] exited with code=0 in 0.647 seconds
```

4. Conditionals (if-else)

```go
package main


import "fmt"


func main() {
    n := -5
    if n > 0 {
        fmt.Println("Positive")
    } else if n < 0 {
        fmt.Println("Negative")
    } else {
        fmt.Println("Zero")
    }

    // Check if a number is even or odd
    num := 7
    if num%2 == 0 {
        fmt.Printf("%d is even\n", num)
    } else {
```

```
        fmt.Printf("%d is odd\n", num)

    }

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t4.go"
Negative
7 is odd

[Done] exited with code=0 in 0.639 seconds
```

5. Loops (for)

```
package main



import "fmt"



func main() {

    for i := 1; i <= 3; i++ {

        fmt.Printf("Iteration %d\n", i)

    }



    counter := 1

    for counter <= 3 {

        fmt.Printf("Counter: %d\n", counter)

        counter++

    }



    // Print numbers from 10 to 1 in reverse

    fmt.Println("Counting backward:")
```

```
    for i := 10; i >= 1; i-- {

        fmt.Printf("%d ", i)

    }

    fmt.Println() // New line after the countdown

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t5.go"
Iteration 1
Iteration 2
Iteration 3
Counter: 1
Counter: 2
Counter: 3
Counting backward:
10 9 8 7 6 5 4 3 2 1

[Done] exited with code=0 in 0.641 seconds
```

6. Basic Functions

```
package main


import "fmt"


func square(x int) int {

    return x * x

}


// Calculate factorial

func factorial(n int) int {

    if n <= 1 {

        return 1
```

```
    }

    return n * factorial(n-1)

}



func main() {

    fmt.Printf("Square of 5: %d\n", square(5))

    fmt.Printf("Factorial of 5: %d\n", factorial(5))

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t6.go"
Square of 5: 25
Factorial of 5: 120

[Done] exited with code=0 in 0.639 seconds
```

7. Multiple Return Values

```
package main



import "fmt"



func swap(a, b string) (string, string) {

    return b, a

}



// Return both sum and product

func sumAndProduct(a, b int) (int, int) {

    return a + b, a * b

}
```

```go
func main() {

    x, y := swap("hello", "world")

    fmt.Println(x, y)



    sum, product := sumAndProduct(5, 3)

    fmt.Printf("Sum: %d, Product: %d\n", sum, product)

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t7.go"
world hello
Sum: 8, Product: 15

[Done] exited with code=0 in 0.64 seconds
```

8. Arrays & Slices

```go
package main



import "fmt"



func main() {

    arr := [3]int{10, 20, 30}

    slice := append(arr[:], 40)

    fmt.Printf("First element: %d\n", arr[0])

    fmt.Println("Slice:", slice)



    // Create and iterate over a slice of strings

    fruits := []string{"Apple", "Orange", "Banana", "Mango"}
```

```
    fmt.Println("Fruits:")

    for i, fruit := range fruits {

        fmt.Printf("%d: %s\n", i, fruit)

    }

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t8.go"
First element: 10
Slice: [10 20 30 40]
Fruits:
0: Apple
1: Orange
2: Banana
3: Mango

[Done] exited with code=0 in 0.639 seconds
```

9. Maps & Structs

```
package main



import "fmt"



type Person struct {

    Name string

    Age  int

}



func main() {

    dict := map[string]string{"name": "Alice", "job": "Engineer"}
```

```
    fmt.Println("Job:", dict["job"])


    // Add a new key-value pair
    dict["location"] = "San Francisco"
    fmt.Println("Location:", dict["location"])


    p := Person{Name: "Alice", Age: 30}
    fmt.Printf("Person: %s, Age: %d\n", p.Name, p.Age)
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t9.go"
Job: Engineer
Location: San Francisco
Person: Alice, Age: 30

[Done] exited with code=0 in 0.635 seconds
```

10. Error Checking

```
package main


import (
    "fmt"
    "strconv"
)


func parseNumber(s string) (int, error) {
    return strconv.Atoi(s)
}
```

```go
func main() {
    if num, err := parseNumber("123"); err == nil {
        fmt.Println("Number:", num)
    } else {
        fmt.Println("Error:", err)
    }

    // Handle invalid input
    if num, err := parseNumber("abc"); err == nil {
        fmt.Println("Number:", num)
    } else {
        fmt.Println("Error:", err)
    }
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\tempCodeRunnerFile.go"
Number: 123
Error: strconv.Atoi: parsing "abc": invalid syntax

[Done] exited with code=0 in 0.272 seconds
```

11. Read/Write Files

```go
package main

import (
    "fmt"
    "os"
```

```
)


func main() {

    os.WriteFile("data.txt", []byte("Go is efficient!"), 0644)

    data, _ := os.ReadFile("data.txt")

    fmt.Println("File content:", string(data))



    // Append a new line to the file

    file, _ := os.OpenFile("data.txt", os.O_APPEND|os.O_WRONLY,
0644)

    defer file.Close()

    file.WriteString("\nGo is also simple and powerful!")



    // Read updated content

    updatedData, _ := os.ReadFile("data.txt")

    fmt.Println("Updated content:", string(updatedData))

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t11.go"
File content: Go is efficient!
Updated content: Go is efficient!
Go is also simple and powerful!

[Done] exited with code=0 in 0.674 seconds
```

12. Goroutines & Channels

```
package main



import (
```

```
    "fmt"
    "time"
)

func printNumbers(ch chan int) {
    for i := 1; i <= 3; i++ {
        ch <- i
        time.Sleep(time.Second)
    }
    close(ch)
}

func sumArray(arr []int, ch chan int) {
    sum := 0
    for _, num := range arr {
        sum += num
    }
    ch <- sum
}

func main() {
    ch := make(chan int)
    go printNumbers(ch)
    for num := range ch {
        fmt.Println("Received:", num)
```

```
    }


    // Create two goroutines to calculate sum of two arrays
    arr1 := []int{1, 2, 3, 4, 5}
    arr2 := []int{6, 7, 8, 9, 10}

    ch1 := make(chan int)
    ch2 := make(chan int)

    go sumArray(arr1, ch1)
    go sumArray(arr2, ch2)

    sum1 := <-ch1
    sum2 := <-ch2

    fmt.Printf("Sum of array1: %d\n", sum1)
    fmt.Printf("Sum of array2: %d\n", sum2)
    fmt.Printf("Total sum: %d\n", sum1+sum2)
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t12.go"
Received: 1
Received: 2
Received: 3
Sum of array1: 15
Sum of array2: 40
Total sum: 55

[Done] exited with code=0 in 3.642 seconds
```

13. Using External Packages (e.g., Gin)

```go
package main


import "github.com/gin-gonic/gin"


func main() {
    r := gin.Default()
    r.GET("/", func(c *gin.Context) {
        c.String(200, "Hello from Gin!")
    })


    // Create a GET endpoint that returns JSON data
    r.GET("/api/data", func(c *gin.Context) {
        c.JSON(200, gin.H{
            "status":  "success",
            "message": "Data retrieved successfully",
            "data": gin.H{
                "id":    123,
                "name":  "Product",
                "price": 29.99,
            },
        })
    })


    r.Run()
```

```
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t13.go"
..\LAB 9 PFAI\t13.go:3:8: no required module provides package github.com/gin-gonic/gin; to add it:
    go get github.com/gin-gonic/gin

[Done] exited with code=1 in 0.097 seconds

[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t13.go"
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /                         --> main.main.func1 (3 handlers)
[GIN-debug] GET    /api/data                 --> main.main.func2 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Environment variable PORT is undefined. Using port :8080 by default
[GIN-debug] Listening and serving HTTP on :8080
```

14. Prime Number Checker

```go
package main



import (

    "fmt"

    "math"

)



// isPrime checks if a number is prime

func isPrime(n int) bool {

    // 0 and 1 are not prime numbers

    if n <= 1 {

        return false

    }



    // 2 is prime
```

```
    if n == 2 {
        return true
    }

    // Even numbers (except 2) are not prime
    if n%2 == 0 {
        return false
    }

    // Check odd divisors up to square root of n
    sqrtN := int(math.Sqrt(float64(n)))
    for i := 3; i <= sqrtN; i += 2 {
        if n%i == 0 {
            return false
        }
    }

    return true
}

func main() {
    testCases := []int{0, 1, 2, 13, 27}

    fmt.Println("Prime Number Checker:")
    for _, num := range testCases {
```

```
        if isPrime(num) {
            fmt.Printf("%d is prime\n", num)
        } else {
            fmt.Printf("%d is not prime\n", num)
        }
    }
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t14.go"
Prime Number Checker:
0 is not prime
1 is not prime
2 is prime
13 is prime
27 is not prime

[Done] exited with code=0 in 0.651 seconds
```

15. Slice Average Calculator

```
package main


import (
    "errors"
    "fmt"
)


// average calculates the mean of a slice of float64 values
func average(numbers []float64) (float64, error) {
    if len(numbers) == 0 {
```

```
        return 0, errors.New("cannot calculate average of empty
slice")
    }

    sum := 0.0
    for _, num := range numbers {
        sum += num
    }

    return sum / float64(len(numbers)), nil
}

func main() {
    fmt.Println("Slice Average Calculator:")

    // Test with non-empty slice
    nums := []float64{5.2, 6.8, 9.1}
    avg, err := average(nums)
    if err == nil {
        fmt.Printf("Average of %v: %.2f\n", nums, avg)
    } else {
        fmt.Printf("Error: %s\n", err)
    }

    // Test with empty slice
```

```
    emptySlice := []float64{}

    avg, err = average(emptySlice)

    if err == nil {

        fmt.Printf("Average of %v: %.2f\n", emptySlice, avg)

    } else {

        fmt.Printf("Error: %s\n", err)

    }

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\tempCodeRunnerFile.go"
Slice Average Calculator:
Average of [5.2 6.8 9.1]: 7.03
Error: cannot calculate average of empty slice

[Done] exited with code=0 in 0.652 seconds
```

16. Employee Management System

```
package main



import "fmt"



// Employee struct with Name, Position, Salary

type Employee struct {

    Name     string

    Position string

    Salary   float64

}



// highestPaid returns the employee with the highest salary
```

```
func highestPaid(employees []Employee) Employee {
    if len(employees) == 0 {
        return Employee{}
    }

    highest := employees[0]
    for _, emp := range employees {
        if emp.Salary > highest.Salary {
            highest = emp
        }
    }

    return highest
}

func main() {
    // Create a slice of 5 employees
    employees := []Employee{
        {Name: "Alice Johnson", Position: "Software Engineer",
Salary: 85000},
        {Name: "Bob Smith", Position: "Project Manager", Salary:
92000},
        {Name: "Carol Davis", Position: "CTO", Salary: 120000},
        {Name: "David Wilson", Position: "UI Designer", Salary:
78000},
```

```
        {Name: "Eva Brown", Position: "DevOps Engineer", Salary:
90000},

    }



    // Find and print the highest paid employee

    topEarner := highestPaid(employees)

    fmt.Printf("Highest paid employee:\n")

    fmt.Printf("Name: %s\n", topEarner.Name)

    fmt.Printf("Position: %s\n", topEarner.Position)

    fmt.Printf("Salary: $%.2f\n", topEarner.Salary)

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t16.go"
Highest paid employee:
Name: Carol Davis
Position: CTO
Salary: $120000.00

[Done] exited with code=0 in 0.642 seconds
```

17. Robust File Appender

```
package main



import (

    "fmt"

    "os"

)



func appendToFile(filename, content string) error {
```

```
    // Check if file exists
    _, err := os.Stat(filename)
    if os.IsNotExist(err) {
        // Create file if it doesn't exist
        return os.WriteFile(filename, []byte(content), 0644)
    }

    // Open file for appending
    file, err := os.OpenFile(filename, os.O_APPEND|os.O_WRONLY,
0644)
    if err != nil {
        return err
    }
    defer file.Close()

    // Append content
    _, err = file.WriteString(content)
    return err
}

func main() {
    filename := "data.txt"
    content := "\nThis is new content appended to the file."

    err := appendToFile(filename, content)
```

```
    if err != nil {
        fmt.Printf("Failed to append to file: %s\n", err)
    } else {
        fmt.Printf("Successfully appended content to %s\n",
filename)

        // Read and print the file content
        data, err := os.ReadFile(filename)
        if err == nil {
            fmt.Printf("Current file content:\n%s\n", string(data))
        }
    }
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\tempCodeRunnerFile.go"
Successfully appended content to data.txt
Current file content:
Go is efficient!
Go is also simple and powerful!
This is new content appended to the file.

[Done] exited with code=0 in 0.691 seconds
```

18. Concurrent Array Sum

```
package main

import "fmt"

func sumArray(arr []int, ch chan int) {
    sum := 0
```

```
    for _, num := range arr {
        sum += num
    }
    ch <- sum
}

func main() {
    // Create two arrays
    array1 := []int{1, 2, 3, 4, 5}
    array2 := []int{6, 7, 8, 9, 10}

    // Create channels for results
    ch1 := make(chan int)
    ch2 := make(chan int)

    // Calculate sum concurrently
    go sumArray(array1, ch1)
    go sumArray(array2, ch2)

    // Receive results
    sum1 := <-ch1
    sum2 := <-ch2

    // Print results
    fmt.Printf("Sum of array1 %v: %d\n", array1, sum1)
```

```
    fmt.Printf("Sum of array2 %v: %d\n", array2, sum2)

    fmt.Printf("Total sum: %d\n", sum1+sum2)

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t18.go"
Sum of array1 [1 2 3 4 5]: 15
Sum of array2 [6 7 8 9 10]: 40
Total sum: 55

[Done] exited with code=0 in 0.643 seconds
```

19. JSON API Endpoint

```
package main


import "github.com/gin-gonic/gin"


func main() {

    r := gin.Default()


    // Create /user endpoint returning JSON

    r.GET("/user", func(c *gin.Context) {

        c.JSON(200, gin.H{

            "id":    123,

            "name":  "Alice",

            "email": "alice@example.com",

        })

    })


    // Start the server
```

```
    r.Run() // Listens on 0.0.0.0:8080

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t19.go"
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /user                     --> main.main.func1 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Environment variable PORT is undefined. Using port :8080 by default
[GIN-debug] Listening and serving HTTP on :8080
```

20. Bank Account Manager

```
package main



import (

    "errors"

    "fmt"

)



// BankAccount struct with balance

type BankAccount struct {

    balance float64

}



// Deposit adds amount to balance

func (a *BankAccount) Deposit(amount float64) error {

    if amount <= 0 {

        return errors.New("deposit amount must be positive")
```

```
    }

    a.balance += amount

    return nil
}


// Withdraw subtracts amount from balance
func (a *BankAccount) Withdraw(amount float64) error {
    if amount <= 0 {
        return errors.New("withdrawal amount must be positive")
    }
    if amount > a.balance {
        return errors.New("insufficient funds")
    }
    a.balance -= amount
    return nil
}


// Balance returns the current balance
func (a *BankAccount) Balance() float64 {
    return a.balance
}


func main() {
    // Create a new account
    account := BankAccount{balance: 100}
```

```
    fmt.Printf("Initial balance: $%.2f\n", account.Balance())

    // Test deposit
    err := account.Deposit(50)
    if err != nil {
        fmt.Printf("Deposit error: %s\n", err)
    } else {
        fmt.Printf("After deposit: $%.2f\n", account.Balance())
    }

    // Test successful withdrawal
    err = account.Withdraw(30)
    if err != nil {
        fmt.Printf("Withdrawal error: %s\n", err)
    } else {
        fmt.Printf("After withdrawal: $%.2f\n", account.Balance())
    }

    // Test overdraft prevention
    err = account.Withdraw(200)
    if err != nil {
        fmt.Printf("Withdrawal error: %s\n", err)
    } else {
        fmt.Printf("After withdrawal: $%.2f\n", account.Balance())
    }
```

```
}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t20.go"
Initial balance: $100.00
After deposit: $150.00
After withdrawal: $120.00
Withdrawal error: insufficient funds

[Done] exited with code=0 in 0.643 seconds
```

21. Shape Interface

```
package main



import (

    "fmt"

    "math"

)



// Shape interface with Area method

type Shape interface {

    Area() float64

}



// Circle implements Shape

type Circle struct {

    radius float64

}



func (c Circle) Area() float64 {
```

```
    return math.Pi * c.radius * c.radius
}

// Rectangle implements Shape
type Rectangle struct {
    width  float64
    height float64
}

func (r Rectangle) Area() float64 {
    return r.width * r.height
}

func main() {
    // Create a slice of different shapes
    shapes := []Shape{
        Circle{radius: 5},
        Rectangle{width: 4, height: 6},
        Circle{radius: 3},
        Rectangle{width: 10, height: 2},
    }

    // Calculate and print each shape's area
    for i, shape := range shapes {
        fmt.Printf("Shape %d area: %.2f\n", i+1, shape.Area())
```

```
    }

}
```

```
[Running] go run "c:\Users\Student\Desktop\LAB 9 PFAI\t21.go"
Shape 1 area: 78.54
Shape 2 area: 24.00
Shape 3 area: 28.27
Shape 4 area: 20.00

[Done] exited with code=0 in 0.654 seconds
```