**Assignment - 5**

Name : Om Kulkarni Date : 10/02/25

Class : TY CSE

PRN : 22510034 Batch : T-3

Source Codes :

// Q1: Define a Book structure and display its details

struct Book {

var title: String

var author: String

var price: Double

var yearPublished: Int

func displayBook() {

print("Title: \(title), Author: \(author), Price: ₹\(price), Year: \(yearPublished)")

}

}

let myBook = Book(title: "Swift Programming", author: "Apple Inc.", price: 599.99, yearPublished: 2021)

myBook.displayBook()

// Q2: Rectangle structure with area calculation

struct Rectangle {

var width: Double

var height: Double

func calcArea() -> Double {

return width \* height

}

}

let myRectangle = Rectangle(width: 10.5, height: 5.2)

print("Rectangle Area: \(myRectangle.calcArea())")

// Q3: Temperature structure with Fahrenheit to Celsius conversion

struct Temperature {

var celsius: Double

init(fahrenheit: Double) {

self.celsius = (fahrenheit - 32) \* 5/9

}

}

let temp = Temperature(fahrenheit: 98.6)

print("Temperature in Celsius: \(temp.celsius)°C")

// Q4: Student structure with default initializer

struct Student {

var name: String

var rollNumber: Int

var marks: Double

init(name: String = "Unknown", rollNumber: Int = 0, marks: Double = 0.0) {

self.name = name

self.rollNumber = rollNumber

self.marks = marks

}

}

let student1 = Student()

let student2 = Student(name: "John Doe", rollNumber: 101, marks: 85.5)

print("Student: \(student2.name), Roll No: \(student2.rollNumber), Marks: \(student2.marks)")

// Q5: Smartphone structure with memberwise initializer

struct Smartphone {

var brand: String

var model: String

var storageGB: Int

var price: Double

}

let myPhone = Smartphone(brand: "Apple", model: "iPhone 15", storageGB: 256, price: 1099.99)

print("Smartphone: \(myPhone.brand) \(myPhone.model), Storage: \(myPhone.storageGB)GB, Price: ₹\(myPhone.price)")

// Q6: BankAccount with minimum balance enforcement

struct BankAccount {

var accountHolder: String

var balance: Double

init(accountHolder: String, balance: Double) {

self.accountHolder = accountHolder

self.balance = balance >= 500 ? balance : 500

}

}

let account1 = BankAccount(accountHolder: "Alice", balance: 1000)

let account2 = BankAccount(accountHolder: "Bob", balance: 300)

print("Account Holder: \(account1.accountHolder), Balance: ₹\(account1.balance)")

print("Account Holder: \(account2.accountHolder), Balance: ₹\(account2.balance)")

// Q7: CarDetails with default year initializer

struct CarDetails {

var brand: String

var model: String

var year: Int

init(brand: String, model: String, year: Int = Calendar.current.component(.year, from: Date())) {

self.brand = brand

self.model = model

self.year = year

}

}

let car1 = CarDetails(brand: "Toyota", model: "Corolla")

let car2 = CarDetails(brand: "Honda", model: "Civic", year: 2019)

print("Car: \(car1.brand) \(car1.model), Year: \(car1.year)")

print("Car: \(car2.brand) \(car2.model), Year: \(car2.year)")

// Q8 & Q9: BankAccount with deposit and withdraw methods

struct BankAccountMutating {

var accountHolder: String

var balance: Double

mutating func deposit(amount: Double) {

balance += amount

}

mutating func withdraw(amount: Double) {

if amount <= balance {

balance -= amount

} else {

print("Insufficient funds!")

}

}

}

var myAccount = BankAccountMutating(accountHolder: "Charlie", balance: 2000)

myAccount.deposit(amount: 500)

myAccount.withdraw(amount: 1000)

print("Final Balance: ₹\(myAccount.balance)")

// Q10: Car with fuel level and mileage

struct Car {

var fuelLevel: Double

var mileage: Double

mutating func refuel(amount: Double) {

fuelLevel = min(fuelLevel + amount, 100)

}

mutating func drive(distance: Double) {

let fuelUsed = distance / mileage

if fuelUsed <= fuelLevel {

fuelLevel -= fuelUsed

} else {

print("Not enough fuel to drive the distance!")

}

}

}

var myCar = Car(fuelLevel: 50, mileage: 15)

myCar.refuel(amount: 30)

myCar.drive(distance: 200)

print("Remaining Fuel: \(myCar.fuelLevel)%")

// Q11: Employee with net salary calculation

struct Employee {

var name: String

var basicSalary: Double

var netSalary: Double {

return basicSalary \* 0.9

}

}

let emp = Employee(name: "David", basicSalary: 50000)

print("Net Salary: ₹\(emp.netSalary)")

// Q12: Speed conversion structure

struct Speed {

var metersPerSecond: Double

var kmPerHour: Double {

return metersPerSecond \* 3.6

}

var milesPerHour: Double {

return metersPerSecond \* 2.237

}

}

let mySpeed = Speed(metersPerSecond: 10)

print("Speed: \(mySpeed.kmPerHour) km/h, \(mySpeed.milesPerHour) mph")

// Q13: CarSpeed with property observers

struct CarSpeed {

var speed: Double {

willSet {

print("Current Speed: \(speed) km/h, Changing to: \(newValue) km/h")

}

didSet {

if speed > 120 {

print("Warning: Speed exceeds 120 km/h!")

}

}

}

}

var myCarSpeed = CarSpeed(speed: 80)

myCarSpeed.speed = 130

// Q14: Copying CarSpeed instance

var anotherCarSpeed = myCarSpeed

anotherCarSpeed.speed = 100

print("Copied Car Speed: \(anotherCarSpeed.speed) km/h")

// Q15: Circle with type property and method

struct Circle {

static let pi: Double = 3.14159

static func area(radius: Double) -> Double {

return pi \* radius \* radius

}

}

print("Circle Area: \(Circle.area(radius: 5.0))")

// Q16: Customer struct with init

struct Customer {

var name: String

var id: Int

init(name: String, id: Int) {

self.name = name

self.id = id

}

}

let customer = Customer(name: "Eve", id: 101)

print("Customer Name: \(customer.name), ID: \(customer.id)")

Output :

