**QUESTION:**

**Utilize the principles of Variables, Control Flow, Arrays, Slices, Maps, and Structs within a specified domain of your choice.**

**Begin the program with a comprehensive overview detailing the scenario and the concepts being implemented.**

**Ensure adequate comments are provided throughout the code. Evaluation of this program will be based on the following criteria:**

**R1: Concept Clarity/Viva: 8 Marks**

**R2: Correctness: 8 Marks**

**R3: Validations( only with if statement expected): 8 Marks**

**R4: Ability to Relate to Real-Time Scenario: 8 Marks**

**R5: Complexity: 8 Marks**

**Scenario:**

The code is based on the domain finance management where a single user is using this application and they are required to enter their bank details, for each bank that is entered, max limit is 10 for a user. The facilities provided are:

* Login
* Adding a bank account
* Display the details of the user
* Changing balance of any existing account

*Structure of the code:*

* The code begins with a package declaration, which specifies the name of the package (main) and the import statements, which are used to import various packages and libraries.
* The code then defines two custom data structures, client and finance, which will be used to store client information and bank details, respectively.
* The readInput function is used to read input from the user. It uses the bufio package to read input from the standard input stream and the strings package to trim any leading or trailing whitespace from the input.
* The menu function is used to display the menu options to the user. It uses the fmt package to print the menu options to the console.
* The main function contains the main logic of the program. It uses a for loop to implement a menu-driven interface, where the user can choose from various options, such as adding details, adding another bank, displaying details, updating the balance, and exiting the program.
* The if statement in the Add Details option checks if the client details have already been entered. If not, it prompts the user to enter their name, age, job role, bank name, current balance, and account type. The input is then stored in the client data structure.
* The if statement in the Add Another bank option checks if the user can add another bank. If so, it prompts the user to enter the bank name, current balance, and account type. The input is then stored in the finance data structure.
* The Display Details option displays the client details and bank details based on the search criteria entered by the user. If the search criteria is "def", the first three bank details are displayed. If the search criteria matches a bank name, the matching bank details are displayed.
* The Update Balance option prompts the user to enter the bank name and the new balance. The new balance is then updated in the finance data structure.
* Overall, the code is well-structured, well-documented, and follows best practices for code formatting and organization.

**CODE:**

package main

import (

"bufio"

"fmt"

"os"

"strings"

)

type client struct {

name string

age int

job string

numberofbanks int

finances [10]finance

}

type finance struct {

balance float64

bankName string

accountType string

}

func readInput() string {

*//to manage problem with input with spaces*

reader := bufio.NewReader(os.Stdin)

input, \_ := reader.ReadString('\n')

return strings.TrimSpace(input)

}

func menu() {

fmt.Println("1. Add Details")

fmt.Println("2. Add Another bank")

fmt.Println("3. Display Details")

fmt.Println("4. Update Balance")

fmt.Println("5. Exit")

}

func main() {

*//variables*

var choice int

var search string

var c client

var newBalance float64

*//function to show menu*

for { *//to mimic do while loop*

menu()

fmt.Print("\nEnter your choice (1/2/3/4/5) : ")

fmt.Scan(&choice)

*//switch case for the menu*

switch choice {

case 1:

if c.name == "" {

fmt.Println("Enter your name : ")

c.name = readInput()

fmt.Println("Enter your age : ")

fmt.Scanln(&c.age)

fmt.Println("Describe your job role : ")

c.job = readInput()

fmt.Println("Enter your Bank name : ")

c.finances[0].bankName = readInput()

fmt.Println("Enter your current balance : ")

fmt.Scanln(&c.finances[0].balance)

fmt.Println("Enter your Account Type : ")

c.finances[0].accountType = readInput()

c.numberofbanks = 1

} else {

fmt.Println("Your data is already present!")

}

case 2:

if c.numberofbanks < 10 {

fmt.Println("Enter your Bank name : ")

fmt.Scanln(&c.finances[c.numberofbanks].bankName)

fmt.Println("Enter your current balance : ")

fmt.Scanln(&c.finances[c.numberofbanks].balance)

fmt.Println("Enter your Account Type : ")

fmt.Scanln(&c.finances[c.numberofbanks].accountType)

c.numberofbanks++

} else {

fmt.Println("You can add only up to 10 banks.")

}

case 3:

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

fmt.Printf("\n %v", c.name)

fmt.Printf("\n \t-%v", c.age)

fmt.Printf("\n \t-%v", c.job)

fmt.Printf("\n You have %v banks entries", c.numberofbanks)

fmt.Println("Enter the name of bank name (press def for default) :")

fmt.Scanln(&search)

fmt.Println("================================")

if search == "def" {

sliced := c.finances[0:3]

for i := 0; i < 3; i++ {

fmt.Printf("\n| %v | %v | %v |", sliced[i].bankName, sliced[i].accountType, sliced[i].balance)

}

} else {

for i := 0; i < c.numberofbanks; i++ {

if c.finances[i].bankName == search {

fmt.Printf("\n %v", c.finances[i].bankName)

fmt.Printf("\n %v", c.finances[i].balance)

fmt.Printf("\n %v", c.finances[i].accountType)

}

}

}

case 4:

fmt.Println("Enter bank name :")

search = readInput()

for i := 0; i < c.numberofbanks; i++ {

if c.finances[i].bankName == search {

fmt.Println("Enter Balance : ")

fmt.Scan(&newBalance)

c.finances[i].balance = newBalance

}

}

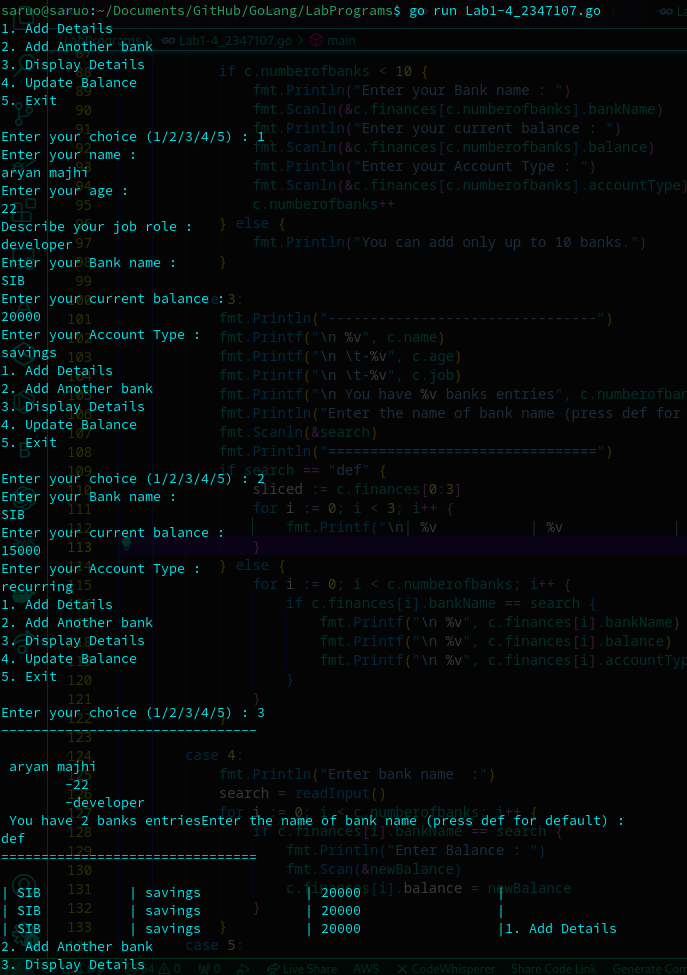
case 5:

return

}

}

}

**Screenshot:**