وظيفة برمجة الشبكات /2

إعداد الطلاب: سوزانة ليلان حمود 2462 غدير حسيب حواط 2486

إشراف الدكتور المهندس:
مهند عيسى

السنة الخامسة
هندسة الاتصالات والالكترونيات

SERVER Code:

```
import socket
import threading

HOST = '0.0.0.0'
PORT = 8989

accounts = {
    "1": {"balance": 1000},
    "2": {"balance": 500}
}

def validate_input(data):
    try:
        account, operation = data.split(",")
        if operation.startswith("check"):
            return account, "check"
        elif operation.startswith("deposit"):
            amount = float(operation.split(" ")[1])
            if amount <= 0:
                return None, "Invalid deposit amount"
            return account, ("deposit", amount)
        elif operation.startswith("withdraw"):
            amount = float(operation.split(" ")[1])
            if amount <= 0:
                return None, "Invalid withdrawal amount"
            return account, ("withdraw", amount)
        else:
            return None, "Invalid request"
    except (ValueError, IndexError):
        return None, "Invalid input format"

def handle_client(conn, addr):
    print(f"Connected by {addr}")
    try:
        while True:
            data = conn.recv(1024).decode().strip()
            if not data:
                break

            account, operation = validate_input(data)
            if operation == "check":
                if account not in accounts:
                    response = "Invalid account!"
```

```
                else:
                    balance = accounts[account]["balance"]
                    response = f"Your balance is: ${balance}"
            elif operation == "deposit":
                if account not in accounts:
                    response = "Invalid account!"
                else:
                    amount = operation[1]
                    accounts[account]["balance"] += amount
                    response = f"Deposited ${amount}. New balance:
${accounts[account]['balance']}"
            elif operation == "withdraw":
                if account not in accounts:
                    response = "Invalid account!"
                else:
                    balance = accounts[account]["balance"]
                    amount = operation[1]
                    if amount > balance:
                        response = "Insufficient funds!"
                    else:
                        accounts[account]["balance"] -= amount
                        response = f"Withdrew ${amount}. New balance:
${accounts[account]['balance']}"
            else:
                response = operation

            conn.sendall(response.encode())
    except Exception as e:
        print(f"Error: {e}")
    finally:
        conn.close()
        print(f"Client {addr} disconnected")

def start_server():
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.bind((HOST, PORT))
            s.listen()
            print(f"Server listening on port {PORT}")
            while True:
                conn, addr = s.accept()
                client_thread = threading.Thread(target=handle_client,
args=(conn, addr))
                client_thread.start()
    except Exception as e:
```

```
        print(f"Error: {e}")

if __name__ == "__main__":
    start_server()
```

CLIENT Code:

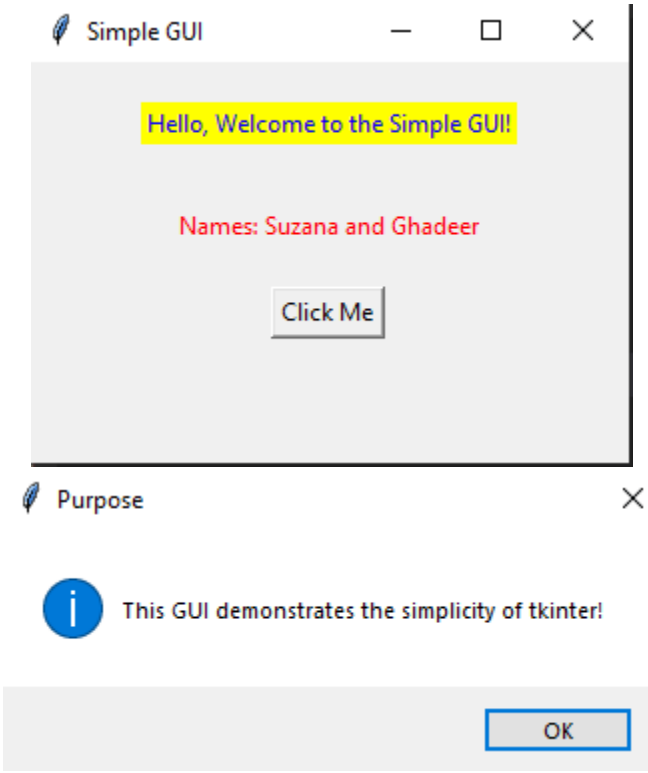
```
import socket

HOST = '127.0.0.1'
PORT = 8989

def validate_input(account, operation, amount=None):
    if not account:
        return "Invalid account name"
    if operation not in ["check", "deposit", "withdraw", "exit"]:
        return "Invalid operation"
    if operation in ["deposit", "withdraw"] and (amount is None or amount
<= 0):
        return "Invalid amount"
    return None

def connect_and_send(account, operation, amount=None):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((HOST, PORT))
            data = f"{account},{operation}"
            if amount:
                data += f" {amount}"
            s.sendall(data.encode())
            response = s.recv(1024).decode()
            print(response)
    except Exception as e:
        print(f"Error: {e}")

def main():
    account = input("Enter account number: ")
    while True:
        operation = input("Enter operation (check, deposit, withdraw,
exit): ")
        if operation == "exit":
            break
        if operation in ["deposit", "withdraw"]:
            try:
```

```
                amount = float(input("Enter amount: "))
            except ValueError:
                print("Invalid amount")
                continue
        else:
            amount = None
        error = validate_input(account, operation, amount)
        if error:
            print(error)
        else:
            connect_and_send(account, operation, amount)

if __name__ == "__main__":
    main()
```

```
import socket

HOST = '127.0.0.2'
PORT = 8989

def validate_input(account, operation, amount=None):
    if not account:
        return "Invalid account name"
    if operation not in ["check", "deposit", "withdraw", "exit"]:
        return "Invalid operation"
    if operation in ["deposit", "withdraw"] and (amount is None or amount
<= 0):
        return "Invalid amount"
    return None

def connect_and_send(account, operation, amount=None):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((HOST, PORT))
            data = f"{account},{operation}"
            if amount:
                data += f" {amount}"
            s.sendall(data.encode())
            response = s.recv(1024).decode()
            print(response)
    except Exception as e:
        print(f"Error: {e}")

def main():
```

```
    account = input("Enter account number: ")
    while True:
        operation = input("Enter operation (check, deposit, withdraw,
exit): ")
        if operation == "exit":
            break
        if operation in ["deposit", "withdraw"]:
            try:
                amount = float(input("Enter amount: "))
            except ValueError:
                print("Invalid amount")
                continue
        else:
            amount = None
        error = validate_input(account, operation, amount)
        if error:
            print(error)
        else:
            connect_and_send(account, operation, amount)

if __name__ == "__main__":
    main()
```

Question2:




```
      import tkinter as tk
from tkinter import messagebox

def display_purpose():
    messagebox.showinfo("Purpose", "This GUI demonstrates the simplicity
of tkinter!")

window = tk.Tk()
window.title("Simple GUI")
window.geometry("300x200")

label = tk.Label(window, text="Hello, Welcome to the Simple GUI!",
fg="blue", bg="yellow")
label.pack(pady=20)

name_label = tk.Label(window, text="Names: Suzana and Ghadeer", fg="red")
name_label.pack(pady=10)

button = tk.Button(window, text="Click Me", command=display_purpose)
button.pack(pady=10)
```

```
window.mainloop()
```