الجمهورية العربية السورية

وزارة التعليم العالي والبحث العلمي

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة

مقرر برمجة الشبكات

Second Network Programming Homework

إعداد الطالبات:

سالي عز الدين سلطان (1483)

ليندا إسماعيل (2686)

راما غياث ناعسة (2539)

إشراف الدكتور المهندس:

مهند عيسى

Question 1: Bank ATM Application with TCP Server/Client and Multi-threading:

```
import socket
import threading
import json

accounts = {
    'Saly': {'password': 'password1', 'balance': 2500},
    'Linda': {'password': 'password2', 'balance': 3500},
    'Rama': {'password': 'password1', 'balance': 2000}
}

def handle_client(client_socket):
    while True:
        received_data = client_socket.recv(1024).decode('utf-8')
        if not received_data:
            break

        data = json.loads(received_data)
        command = data['command']
        username = data['username']
        password = data['password']

        if username in accounts and accounts[username]['password'] ==
password:
            if command == 'balance':
                response = {'status': 'success', 'balance':
accounts[username]['balance']}
            elif command == 'deposit':
                amount = data['amount']
                accounts[username]['balance'] += amount
                response = {'status': 'success', 'balance':
accounts[username]['balance']}
            elif command == 'withdraw':
                amount = data['amount']
                if accounts[username]['balance'] >= amount:
                    accounts[username]['balance'] -= amount
                    response = {'status': 'success', 'balance':
accounts[username]['balance']}
                else:
                    response = {'status': 'error', 'message':
'Insufficient funds'}
            else:
                response = {'status': 'error', 'message': 'Invalid
command'}
```

```
        else:
            response = {'status': 'error', 'message': 'Authentication
failed'}

        client_socket.send(json.dumps(response).encode('utf-8'))

    client_socket.close()

def start_server():
    server_ip = '0.0.0.0'
    server_port = 65432

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))
    server_socket.listen()

    print(f'Server is listening on {server_ip}:{server_port}')

    while True:
        client_socket, client_address = server_socket.accept()
        print(f'Accepted connection from {client_address}')
        client_handler = threading.Thread(target=handle_client,
args=(client_socket,))
        client_handler.start()

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

التنفيذ:

```
Server is listening on 0.0.0.0:6666
Accepted connection from ('127.0.0.1', 15754)
Accepted connection from ('127.0.0.1', 15838)
```

```
Enter username: Saly
Enter password: password1
Enter command (balance/deposit/withdraw/exit): balance
Balance: 2500
Enter command (balance/deposit/withdraw/exit): exit
```

```python
import socket
import json

def bank_client():
    server_ip = '127.0.0.1'
    server_port = 6666

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))

    # User authentication
    username = input('Enter username: ')
    password = input('Enter password: ')

    # Banking operations
    while True:
        command = input('Enter command (balance/deposit/withdraw/exit): ')
        if command in ['deposit', 'withdraw']:
            amount = float(input('Enter amount: '))
        else:
            amount = 0

        # Send command to the server
        data = {
            'command': command,
            'username': username,
            'password': password,
            'amount': amount
        }
        client_socket.send(json.dumps(data).encode('utf-8'))

        # Receive response from the server
        response = json.loads(client_socket.recv(1024).decode('utf-8'))
        if response['status'] == 'success':
            print(f"Balance: {response['balance']}")
        else:
            print(f"Error: {response['message']}")

        if command == 'exit':
            break

    client_socket.close()

if __name__ == '__main__':
    bank_client()
```

```python
import socket
import json

def bank_client():
    server_ip = '127.0.0.2'
    server_port = 6666

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))

    # User authentication
    username = input('Enter username: ')
    password = input('Enter password: ')

    # Banking operations
    while True:
        command = input('Enter command (balance/deposit/withdraw/exit): ')
        if command in ['deposit', 'withdraw']:
            amount = float(input('Enter amount: '))
        else:
            amount = 0

        # Send command to the server
        data = {
            'command': command,
            'username': username,
            'password': password,
            'amount': amount
        }
        client_socket.send(json.dumps(data).encode('utf-8'))

        # Receive response from the server
        response = json.loads(client_socket.recv(1024).decode('utf-8'))
        if response['status'] == 'success':
            print(f"Balance: {response['balance']}")
        else:
            print(f"Error: {response['message']}")

        if command == 'exit':
            break

    client_socket.close()

if __name__ == '__main__':
    bank_client()
```

```python
import socket
import json

def bank_client():
    server_ip = '127.0.0.3'
    server_port = 6666

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))

    # User authentication
    username = input('Enter username: ')
    password = input('Enter password: ')

    # Banking operations
    while True:
        command = input('Enter command (balance/deposit/withdraw/exit): ')
        if command in ['deposit', 'withdraw']:
            amount = float(input('Enter amount: '))
        else:
            amount = 0

        # Send command to the server
        data = {
            'command': command,
            'username': username,
            'password': password,
            'amount': amount
        }
        client_socket.send(json.dumps(data).encode('utf-8'))

        # Receive response from the server
        response = json.loads(client_socket.recv(1024).decode('utf-8'))
        if response['status'] == 'success':
            print(f"Balance: {response['balance']}")
        else:
            print(f"Error: {response['message']}")

        if command == 'exit':
            break

    client_socket.close()

if __name__ == '__main__':
    bank_client()
```

Question 2: Tic Tac Toe using tkinter:

```
import tkinter as tk
from tkinter import font

def print_board(board):
    for i in range(3):
        for j in range(3):
            if board[i][j] == "X":
                buttons[i][j].config(text="X", fg="blue", font=large_font)
            elif board[i][j] == "O":
                buttons[i][j].config(text="O", fg="red", font=large_font)
            else:
                buttons[i][j].config(text="", font=large_font)

def check_winner(board, player):
    for i in range(3):
        if board[i][0] == player and board[i][1] == player and board[i][2]
== player:
            return True
    for i in range(3):
        if board[0][i] == player and board[1][i] == player and board[2][i]
== player:
            return True
    if board[0][0] == player and board[1][1] == player and board[2][2] ==
player:
        return True
    if board[0][2] == player and board[1][1] == player and board[2][0] ==
player:
        return True
    return False

def handle_click(row, col):
    global current_player
    if board[row][col] == " ":
        board[row][col] = current_player
        print_board(board)
        if check_winner(board, current_player):
            if current_player == "X":
                status_label.config(text="Player X wins!", fg="blue")
            else:
                status_label.config(text="Player O wins!", fg="red")
            disable_buttons()
            return
        if current_player == "X":
```

```
            current_player = "O"
        else:
            current_player = "X"
        status_label.config(text=f"Player {current_player}'s turn",
fg="black")

def disable_buttons():
    for row in buttons:
        for button in row:
            button.config(state="disabled")

board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]

current_player = "X"

root = tk.Tk()
root.title("Tic Tac Toe")

large_font = font.Font(size=24, weight="bold")

buttons = []
for i in range(3):
    row = []
    for j in range(3):
        button = tk.Button(root, text="", width=5, height=2,
command=lambda x=i, y=j: handle_click(x, y))
        button.grid(row=i, column=j, padx=10, pady=10)
        row.append(button)
    buttons.append(row)

status_label = tk.Label(root, text=f"Player {current_player}'s turn",
font=large_font, pady=10)
status_label.grid(row=3, column=0, columnspan=3)

root.mainloop()
```

1) import tkinter as tk: Imports the Tkinter module and renames it as tk for easier reference.

2) from tkinter import font: Imports the font submodule from Tkinter to handle font properties.

3) def print_board(board): Defines a function print_board to update the GUI with the current state of the game board.

4) def check_winner(board, player): Defines a function check_winner to determine if a player has won the game.

5) def handle_click(row, col): Defines a function handle_click to handle user clicks on the game board buttons.
6) def disable_buttons(): Defines a function disable_buttons to disable all game board buttons when the game ends.

7) board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]: Initializes the game board with empty spaces.

8) current_player = "X": Initializes the current player as "X."

9) root = tk.Tk(): Creates the main Tkinter window.

10) root.title("Tic Tac Toe"): Sets the title of the window to "Tic Tac Toe."

11) large_font = font.Font(size=24, weight="bold"): Creates a larger font for the game display.

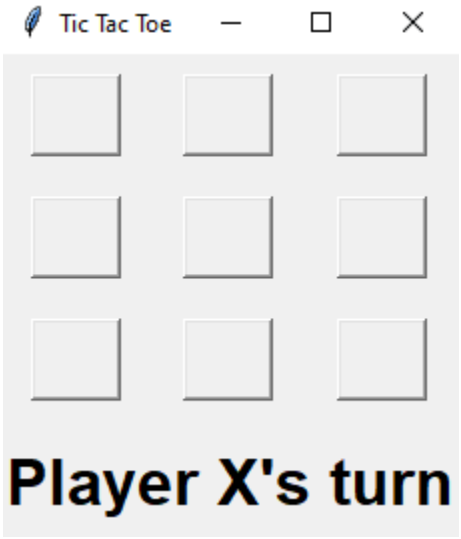
12) buttons = []: Initializes an empty list to store the game board buttons.

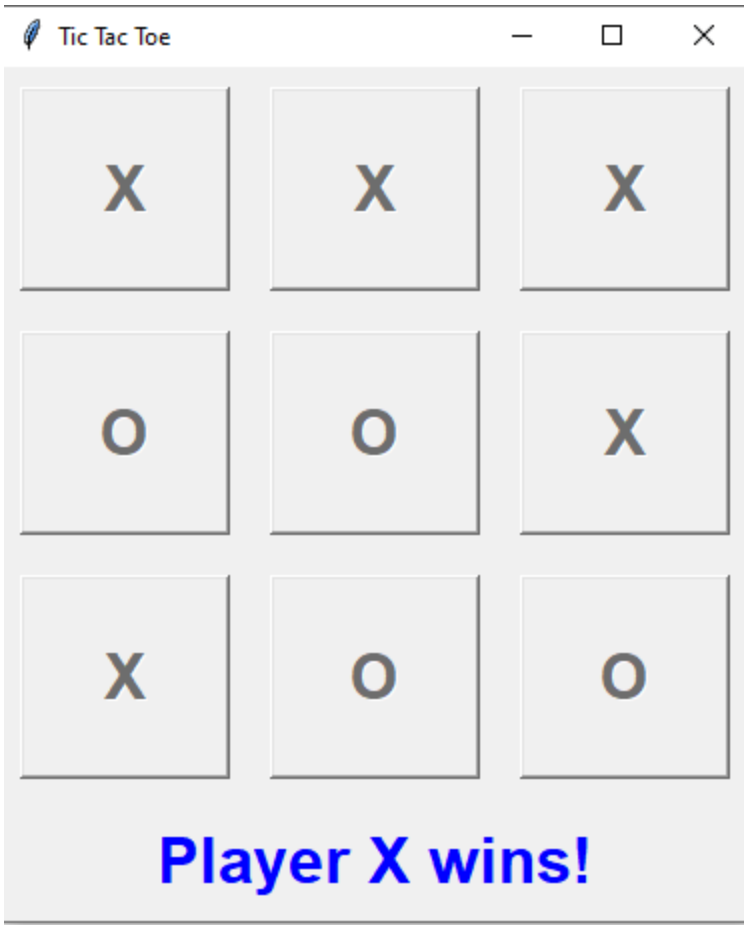
13) Nested loops create a 3x3 grid of buttons representing the game board.

14) status_label = tk.Label(root, text=f"Player {current_player}'s turn", font=large_font, pady=10): Creates a label to display the current player's turn.

15) status_label.grid(row=3, column=0, columnspan=3): Places the status label at the bottom of the window.

16) root.mainloop(): Starts the Tkinter event loop to display the window and handle user interactions.




Tic Tac Toe
X
X
X
O
O
X
X
O
O
Player X wins!