ASSIGNMENT-4

HALL TICKET NO:2303A51773

Objective

To design, develop, and deploy a simple ERC20 Token smart contract using Solidity, following Ethereum standards.

wallet blockchain.py

C: > Users > RANJITH > OneDrive > Desktop > Block chain > wallet blockchain.py > ...

```python
import tkinter as tk
from tkinter import messagebox

class MyERC20Token:
    def __init__(self, initial_supply, name="MyToken", symbol="MTK", decimals=18):
        self.name = name
        self.symbol = symbol
        self.decimals = decimals
        self.total_supply = initial_supply * (10 ** decimals)
        self.balance_of = {}
        self.allowance = {}
        self.owner = "owner"
        self.balance_of[self.owner] = self.total_supply

    def transfer(self, sender, to, value):
        if self.balance_of.get(sender, 0) < value:
            raise Exception("Insufficient balance")
        self.balance_of[sender] -= value
        self.balance_of[to] = self.balance_of.get(to, 0) + value
        return f"Transfer: {value} {self.symbol} from {sender} to {to}"

    def approve(self, owner, spender, value):
        if owner not in self.allowance:
            self.allowance[owner] = {}
        self.allowance[owner][spender] = value
        return f"Approval: {spender} can spend {value} {self.symbol} from {owner}"

    def transfer_from(self, spender, from_addr, to, value):
        if self.balance_of.get(from_addr, 0) < value:
            raise Exception("Insufficient balance")
        if self.allowance.get(from_addr, {}).get(spender, 0) < value:
            raise Exception("Allowance exceeded")
        self.balance_of[from_addr] -= value
        self.balance_of[to] = self.balance_of.get(to, 0) + value
        self.allowance[from_addr][spender] -= value
        return f"TransferFrom: {value} {self.symbol} from {from_addr} to {to} by {spender}"
```

```python
class TokenApp:
    def __init__(self, root, token):
        self.token = token
        self.root = root
        self.root.title("MyToken (MTK) - ERC20 Demo")
        self.root.configure(bg="#1e1e2f")  # Dark background

        # Banner-style header
        header = tk.Label(root, text="APPLICATION STARTED",
                          font=("Courier", 16, "bold"),
                          fg="cyan", bg="#1e1e2f")
        header.pack(fill="x", pady=5)

        # Info section
        info = f"""
Constructor Executed:
• Token initialized as: {token.name} ({token.symbol})

State Variables:
• name → {token.name}
• symbol → {token.symbol}
• decimals → {token.decimals}
• totalSupply → {token.total_supply}

Functions Available:
• transfer() → Send tokens
• approve() → Authorize spender
• transferFrom() → Transfer via allowance
"""
        tk.Label(root, text=info, font=("Courier", 10),
                 justify="left", bg="#2e2e3f", fg="white").pack(fill="x", padx=10, pady=5)

        # Entry fields with labels
        tk.Label(root, text="Sender:", fg="yellow", bg="#1e1e2f").pack()
        self.sender_entry = tk.Entry(root, bg="#333", fg="white")
        self.sender_entry.insert(0, "owner")
```

```
        self.sender_entry.insert(0, "owner")
        self.sender_entry.pack()

        tk.Label(root, text="Receiver:", fg="yellow", bg="#1e1e2f").pack()
        self.receiver_entry = tk.Entry(root, bg="#333", fg="white")
        self.receiver_entry.insert(0, "Alice")
        self.receiver_entry.pack()

        tk.Label(root, text="Amount:", fg="yellow", bg="#1e1e2f").pack()
        self.amount_entry = tk.Entry(root, bg="#333", fg="white")
        self.amount_entry.insert(0, "100")
        self.amount_entry.pack()

        # Buttons with custom colors
        tk.Button(root, text="Transfer", command=self.transfer,
                  bg="green", fg="white").pack(pady=5)
        tk.Button(root, text="Approve", command=self.approve,
                  bg="blue", fg="white").pack(pady=5)
        tk.Button(root, text="Transfer From", command=self.transfer_from,
                  bg="purple", fg="white").pack(pady=5)

        # Output area
        self.output = tk.Text(root, height=10, width=70, bg="#111", fg="lime")
        self.output.pack(pady=10)

    def transfer(self):
        try:
            sender = self.sender_entry.get()
            receiver = self.receiver_entry.get()
            amount = int(self.amount_entry.get())
            result = self.token.transfer(sender, receiver, amount)
            self.output.insert("end", result + "\n")
        except Exception as e:
            messagebox.showerror("Error", str(e))
```

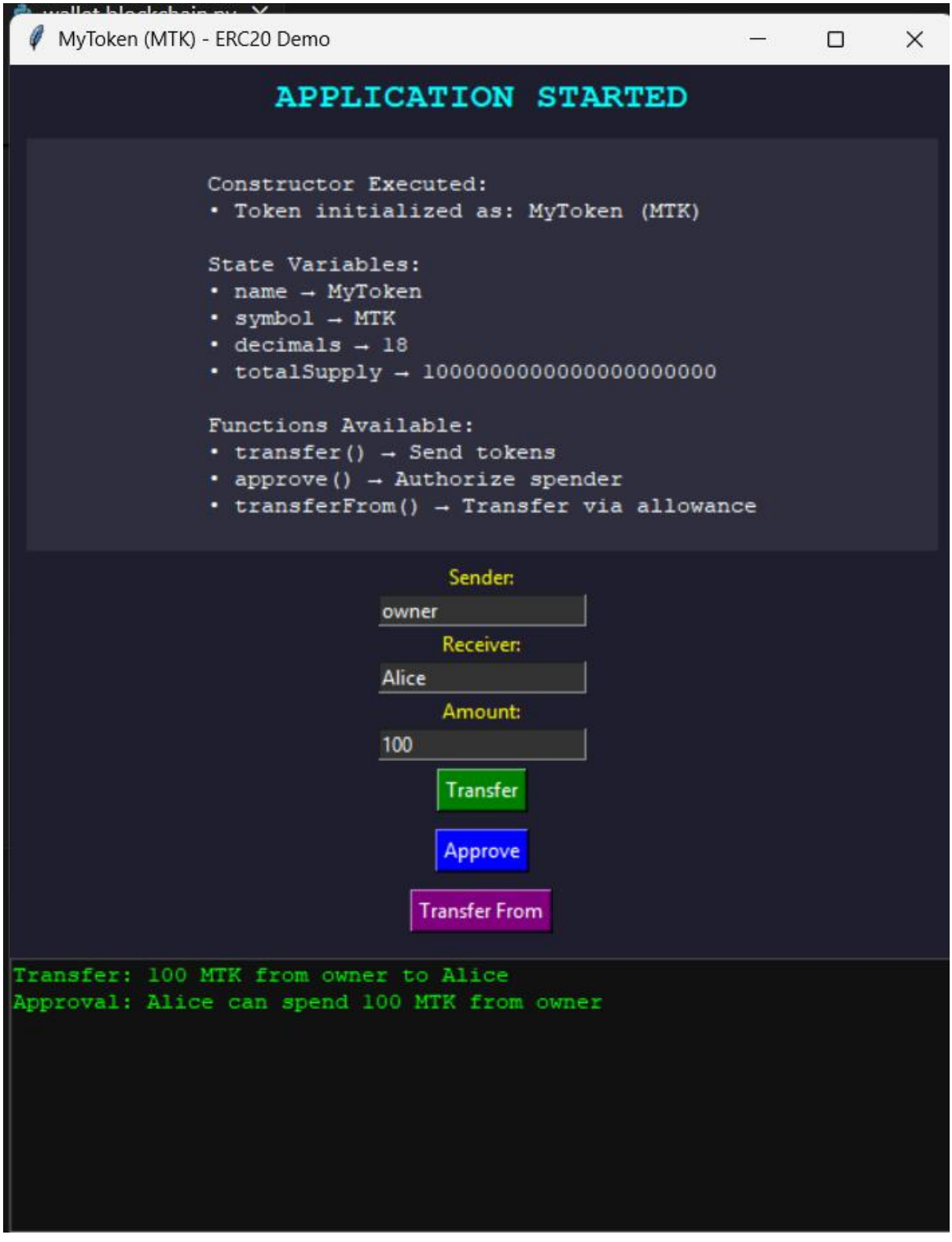
```
            self.output.insert("end", result + "\n")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def approve(self):
        try:
            owner = self.sender_entry.get()
            spender = self.receiver_entry.get()
            amount = int(self.amount_entry.get())
            result = self.token.approve(owner, spender, amount)
            self.output.insert("end", result + "\n")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def transfer_from(self):
        try:
            spender = self.sender_entry.get()
            from_addr = "Alice"  # fixed for demo
            to = self.receiver_entry.get()
            amount = int(self.amount_entry.get())
            result = self.token.transfer_from(spender, from_addr, to, amount)
            self.output.insert("end", result + "\n")
        except Exception as e:
            messagebox.showerror("Error", str(e))


if __name__ == "__main__":
    root = tk.Tk()
    token = MyERC20Token(initial_supply=1000)
    app = TokenApp(root, token)
    root.mainloop()
```

Output:


MyToken (MTK) - ERC20 Demo
APPLICATION STARTED
Constructor Executed:
• Token initialized as: MyToken (MTK)
State Variables:
• name → MyToken
• symbol → MTK
• decimals → 18
• totalSupply → 100000000000000000000
Functions Available:
• transfer() → Send tokens
• approve() → Authorize spender
• transferFrom() → Transfer via allowance
Sender:
owner
Receiver:
Alice
Amount:
100
Transfer
Approve
Transfer From
Transfer: 100 MTK from owner to Alice
Approval: Alice can spend 100 MTK from owner