def dfa\_ends\_with\_ab(input\_string):

state = 0

for char in input\_string:

if state == 0 and char == 'a':

state = 1

elif state == 1 and char == 'b':

state = 0

elif state == 1 and char == 'a':

state = 1

else:

state = 0

return input\_string.endswith("ab")

def test\_dfa():

print("\n=== Testing DFA for strings ending with 'ab' ===")

test\_strings = ["ab", "aab", "abb", "abab", "a", "b", "ba", ""]

for s in test\_strings:

result = "Accepted" if dfa\_ends\_with\_ab(s) else "Rejected"

print(f"String '{s}': {result}")

def nfa\_contains\_ab(input\_string):

current\_states = {0}

for char in input\_string:

next\_states = set()

for state in current\_states:

if state == 0:

if char == 'a':

next\_states.add(0)

next\_states.add(1)

elif char == 'b':

next\_states.add(0)

elif state == 1:

if char == 'a':

pass

elif char == 'b':

next\_states.add(2)

elif state == 2:

next\_states.add(2)

current\_states = next\_states

return 2 in current\_states

def test\_nfa():

print("\n=== Testing NFA for strings containing 'ab' ===")

test\_strings = ["ab", "ba", "aaba", "bab", "a", "b", ""]

for s in test\_strings:

result = "Accepted" if nfa\_contains\_ab(s) else "Rejected"

print(f"String '{s}': {result}")

class VendingMachine:

def \_\_init\_\_(self):

self.state = 0

self.money\_inserted = 0

self.log = []

def insert\_coin(self, coin=25):

if coin != 25:

self.log.append(f"Machine only accepts 25¢ coins. Returned {coin}¢")

return

self.money\_inserted += coin

if self.state == 0:

self.state = 1

self.log.append("Inserted 25¢. Total: 25¢")

elif self.state == 1:

self.state = 2

self.log.append("Inserted 25¢. Total: 50¢")

elif self.state == 2:

self.state = 3

self.log.append("Inserted 25¢. Total: 75¢")

elif self.state == 3:

self.state = 0

self.money\_inserted = 0

self.log.append("Inserted 25¢. Total: 100¢")

self.log.append("Item dispensed! Thank you.")

def cancel(self):

if self.money\_inserted > 0:

self.log.append(f"Transaction cancelled. Returned {self.money\_inserted}¢")

self.money\_inserted = 0

self.state = 0

else:

self.log.append("No money to return.")

def get\_status(self):

state\_names = ["0¢", "25¢", "50¢", "75¢"]

return f"Current state: {state\_names[self.state]}, Money inserted: {self.money\_inserted}¢"

def get\_log(self):

return self.log

def test\_fsm():

print("\n=== Testing Vending Machine FSM ===")

print("Scenario 1: Insert enough coins to get an item")

vm1 = VendingMachine()

print(f"Initial: {vm1.get\_status()}")

vm1.insert\_coin()

vm1.insert\_coin()

vm1.insert\_coin()

vm1.insert\_coin()

print("\n".join(vm1.get\_log()))

print(f"Final: {vm1.get\_status()}")

print("\nScenario 2: Insert some coins and cancel")

vm2 = VendingMachine()

print(f"Initial: {vm2.get\_status()}")

vm2.insert\_coin()

vm2.insert\_coin()

vm2.cancel()

print("\n".join(vm2.get\_log()))

print(f"Final: {vm2.get\_status()}")

print("\nScenario 3: Various actions")

vm3 = VendingMachine()

print(f"Initial: {vm3.get\_status()}")

vm3.insert\_coin()

vm3.insert\_coin(10)

vm3.insert\_coin()

vm3.insert\_coin()

vm3.cancel()

vm3.insert\_coin()

print("\n".join(vm3.get\_log()))

print(f"Final: {vm3.get\_status()}")

if \_\_name\_\_ == "\_\_main\_\_":

test\_dfa()

test\_nfa()

test\_fsm()