Nama:

- Elvina Kartika Aini (23/517213/PA/22161)
- Salmaa Ceiba Abdillah (23/521206/PA/22409)

Implementasi DFA Vending Machine

1. Deskripsi Permasalahan

Vending Machine (Mesin Penjual Minuman Kaleng) adalah mesin otomatis yang menjual minuman dengan menerima input nominal uang, kemudian memprosesnya melalui beberapa state dan menentukan apakah suatu minuman dapat dibeli atau tidak. Pada tugas ini, kami mensimulasikan vending machine tersebut dengan mengimplementasikan DFA (Deterministic Finite Automata) untuk mengetahui bagaimana transisi state dari mesin tersebut.

Vending Machine yang disimulasikan memiliki spesifikasi sebagai berikut :

- Mesin menerima input uang dengan nominal pecahan Rp1.000, Rp2.000, Rp5.000 dan Rp10.000. Selain nominal pecahan tersebut, mesin akan menolak input uang.
- Mesin hanya menerima maksimal nominal uang yang diinputkan adalah Rp10.000. Apabila jumlah nominal uang yang diinputkan melebihi batas tersebut, mesin akan menolak input uang.
- Mesin menjual tiga minuman dengan masing-masing harga sebagai berikut : Minuman A (Rp3.000), Minuman B (Rp4.000), Minuman C (Rp6.000). Apabila input uang sesuai dengaan harga minuman/lebih, maka tombol ON pada minuman akan menyala.
- Mesin juga mencatat jumlah saldo yang sudah diinputkan pengguna, untuk memudahkan transaksi.
- Apabila input uang kurang dari harga minuman ataupun melebihi harga minuman yang tertera, mesin akan menolak transaksi. Dikarenakan tidak ada fitur kembalian pada mesin.
- Mesin juga mencatat lintasan state DFA yang dilalui setiap melakukan transaksi, baik itu transaksi yang berhasil atau ditolak.

Kemudian, kami juga menambahkan fitur kembalian untuk vending machine tersebut. Dimana jika input nominal uang sesuai/lebih dari harga minuman, mesin masih dapat memproses transaksi (transaksi tidak ditolak). Mesin juga dapat mengembalikan kembalian uang apabila input uang melebihi jumlah harga minuman yang dibeli. Mesin hanya dapat melakukan satu transaksi per minuman, apabila minuman yang dipilih sudah diproses dan berhasil, mesin akan di reset ke state awal, meskipun kembalian uang masih cukup untuk membeli minuman lain. Artinya, mesin tidak dapat membeli 2 atau lebih minuman dalam satu kali transaksi.

2. Notasi DFA lengkap

- States pada Vending Machine
  State-state yang ada pada vending machine bertotal 11 state, yaitu : S0, S1000, S2000, S3000, S4000, S5000, S6000, S7000, S8000, S9000, S10000.

- Alphabet
  Alphabet yang diterima pada DFA nya adalah nominal input uang, yaitu : 1000, 2000, 5000, dan 10000. Selain dari nominal uang tersebut, mesin akan menolak uang. Juga untuk input minuman itu sendiri, mesin menerima input huruf kapital, yaitu : A, B, dan C.

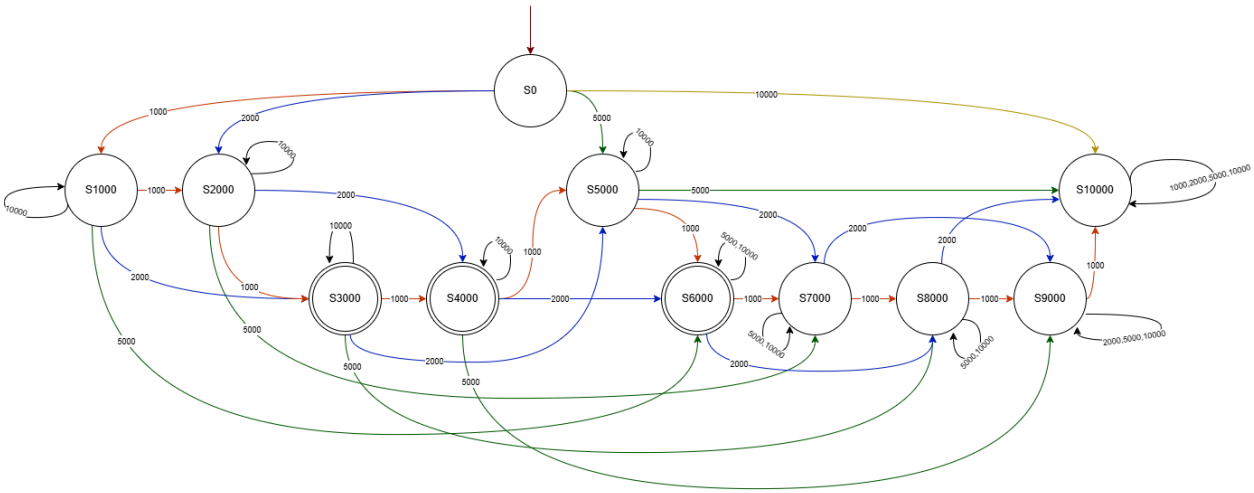
- Start State
  Start state dimulai dari S0.
  Apabila mesin selesai melakukan satu transaksi, mesin direset dan kembali ke state S0.

- Accepting States
  Accepting state berada pada 3 state, yaitu : S3000, S4000, dan S6000. Untuk mesin tanpa fitur kembalian. Untuk mesin dengan fitur kembalian, accepting statenya berada pada state-state yang jumlah input uang melebihi harga minuman tersebut.

- Tabel transisi:

STATE\INPUT	1000	2000	5000	10000
S0	S1000	S2000	S5000	S10000
S1000	S2000	S3000	S6000	S1000
S2000	S3000	S4000	S7000	S2000
S3000*	S4000	S5000	S8000	S3000
S4000*	S5000	S6000	S9000	S4000
S5000	S6000	S7000	S10000	S5000
S6000*	S7000	S8000	S6000	S6000
S7000	S8000	S9000	S7000	S7000
S8000	S9000	S10000	S8000	S8000
S9000	S10000	S9000	S9000	S9000
S10000	S10000	S10000	S10000	S10000

- State Diagram:



3. Penjelasan implementasi program

1) Import modul

```
import re
```

Mengimpor modul re yang digunakan untuk menangani regular expression dalam membaca file transisi.

2) Kelas VendingMachineDFA

```
class VendingMachineDFA:
```

Kelas ini merepresentasikan mesin DFA (Deterministic Finite Automaton) untuk vending machine.

a) Konstruktor (__init__)

```
def __init__(self, transition_file, refund_enabled=False):
```

- transition_file: Nama file yang berisi daftar transisi DFA.
- refund_enabled: Opsi untuk mengaktifkan fitur pengembalian uang (saat ini tidak digunakan).

b) Inisialisasi atribut

```
self.states = set()
self.alphabet = set()
self.start_state = "S0"
self.accept_states = set()
self.reject_states = "REJECT"
self.transitions = {}
self.current_state = self.start_state
self.balance = 0
self.refund_enabled = refund_enabled
self.trace = [self.current_state]
self.load_transitions(transition_file)
```

- self.states: Menyimpan semua state dalam DFA.
- self.alphabet: Menyimpan daftar input uang yang diterima.
- self.start_state: State awal DFA (S0).
- self.accept_states: State akhir yang diterima.
- self.reject_states: State penolakan (REJECT).
- self.transitions: Menyimpan aturan transisi dari satu state ke state lain.
- self.current_state: Menyimpan state saat ini.
- self.balance: Menyimpan saldo pengguna.
- self.trace: Menyimpan lintasan DFA dari awal hingga akhir.
- Memanggil fungsi load_transitions untuk membaca file transisi.

3) Fungsi load_transitions

```
def load_transitions(self, filename):
```

Membaca file konfigurasi yang berisi daftar state dan transisi DFA.

a) Membaca file

```
with open(filename, "r") as file:
    lines = file.readlines()
```

Membuka file transisi dalam mode baca.

b) Regular expression untuk transisi

```
transition_pattern =
re.compile(r"(S\d+)\s+(\d+)\s+(S\d+\*?)")
```

Mencocokkan pola transisi.

c) Memroses setiap baris

```
        for line in lines:
            line = line.strip()
            if line.startswith("States:"):
                self.states = set(line.split(": ")[1].split(",
"))
            elif line.startswith("Alphabet:"):
                self.alphabet = set(map(int, line.split(":
")[1].split(", ")))
            elif line.startswith("Start:"):
                self.start_state = line.split(": ")[1]
                self.current_state = self.start_state
            elif line.startswith("Accept:"):
                self.accept_states = set(line.split(":
")[1].split(", "))
```

Membaca daftar state, alfabet (nominal uang yang diterima), start state, dan accept state.

d) Menyimpan transisi (lintasan)

```
            elif line and not line.startswith("Transitions:"):
                match = transition_pattern.match(line)
                if match:
                    state_from, amount, state_to = match.groups()
                    amount = int(amount)
                    if state_to.endswith("*"):
                        state_to = state_to.rstrip("*")
                        self.accept_states.add(state_to)
                    if state_from not in self.transitions:
                        self.transitions[state_from] = {}
                    self.transitions[state_from][amount] =
state_to
```

Menyimpan aturan transisi antara state berdasarkan input nominal.

4) Fungsi insert_money

```
    def insert_money(self, amount):
```

Menangani input uang dari pengguna.

a) Validasi input

```
        if amount not in self.alphabet:
            print("Nominal tidak diterima! Harap masukkan pecahan
1000, 2000, 5000 atau 10000")
            return
        if self.balance + amount > 10000:
            print("Batas maksimal uang yang diterima adalah
Rp10.000!")
            return
```

Menolak uang yang tidak valid atau melebihi Rp10.000,00.

b) Memperbarui saldo dan state

```
        self.balance += amount
        next_state = self.transitions.get(self.current_state,
{}).get(amount, self.current_state)

        if next_state != self.current_state:
            self.current_state = next_state
            self.trace.append(self.current_state)
```

Menambah saldo dan memperbarui state sesuai aturan transisi.

c) Menampilkan saldo dan minuman tersedia

```
        available_drinks = self.get_available_drinks()
        if available_drinks:
            print(f"ON: {', '.join(available_drinks)}")
        print(f"Saldo saat ini: Rp{self.balance}")
```

Menampilkan saldo dan daftar minuman yang dapat dibeli.

5) Fungsi get_available_drinks

```
    def get_available_drinks(self):
        available = []
        if self.balance >= 3000: available.append("Minuman A")
```

```
        if self.balance >= 4000: available.append("Minuman B")
        if self.balance >= 6000: available.append("Minuman C")
        return available
```

Mengembalikan daftar minuman yang bisa dibeli berdasarkan saldo.

6) Fungsi buy_drink

```
    def buy_drink(self, choice):
```

Menangani pembelian minuman.

a) Menentukan harga minuman

```
        price_map = {"A": 3000, "B": 4000, "C": 6000}
```

Harga masing-masing minuman.

b) Validasi pembelian

```
        if choice in price_map and self.balance == price_map[choice]:
            print(f"Lintasan DFA: {' → '.join(self.trace)}")
            print(f"Minuman {choice} dapat dibeli. Status:
ACCEPTED.")
            print("Terima kasih telah menggunakan vending machine!")
            exit()
```

Jika saldo pas, transaksi diterima, dan program selesai.

c) Menolak transaksi tidak valid

```
        else:
            self.current_state = self.reject_states
            self.trace.append(self.current_state)
            print(f"Lintasan DFA: {' → '.join(self.trace)}")
            print("Saldo tidak pas. Status: REJECTED")
            exit()
```

Jika saldo kurang atau lebih, transaksi ditolak.

7) Loop pengguna

```
vending_machine = VendingMachineDFA("vending_dfa.txt",
refund_enabled=False)
```

Membuat objek vending machine dengan aturan dari file vending_dfa.txt.

```
while True:
    user_input = input("Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): ")
    if user_input in {"A", "B", "C"}:
        vending_machine.buy_drink(user_input)
    else:
        try:
            vending_machine.insert_money(int(user_input))
        except ValueError:
            print("Input tidak valid, masukkan nominal angka atau pilihan minuman!")
```

Loop berjalan terus hingga pengguna membeli minuman atau keluar dari program.

4. Contoh masukan dan keluaran

- Berikut contoh masukan dan keluaran pada program vending_machine.py (tanpa fitur pengembalian)

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Saldo saat ini: Rp1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 2000
ON: Minuman A
Saldo saat ini: Rp3000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): A
Lintasan DFA: S0 → S1000 → S3000
Minuman A dapat dibeli. Status: ACCEPTED.
Terima kasih telah menggunakan vending machine!
```

Masukan uang sesuai dengan harga minuman

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, Minuman B
Saldo saat ini: Rp5000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): B
Lintasan DFA: S0 → S5000 → REJECT
Saldo tidak pas. Status: REJECTED
```

Masukan uang tidak sesuai (melebihi) harga minuman

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Saldo saat ini: Rp1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 2000
ON: Minuman A
Saldo saat ini: Rp3000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): B
Lintasan DFA: S0 → S1000 → S3000 → REJECT
Saldo tidak pas. Status: REJECTED
```

Masukan uang tidak sesuai (kurang dari) harga minuman

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, Minuman B
Saldo saat ini: Rp5000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, Minuman B, Minuman C
Saldo saat ini: Rp10000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
Batas maksimal uang yang diterima adalah Rp10.000!
```

Masukan uang melebihi nominal maksimal (Rp10.000,00)

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 100
Nominal tidak diterima! Harap masukkan pecahan 1000, 2000, 5000 atau 10000
```

Masukan uang tidak sesuai nominal yang diterima

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): H
Input tidak valid, masukkan nominal angka atau pilihan minuman!
```

Masukan tidak sesuai nominal yang diterima, maupun jenis minuman yang tersedia

- Berikut contoh masukan dan keluaran pada program vending_machine_withRefund.py (dengan fitur pengembalian)

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 1000
Saldo saat ini: Rp1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 2000
ON: Minuman A
Saldo saat ini: Rp3000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): A
Lintasan DFA: S0 → S1000 → S3000
Minuman A dapat dibeli. Status: ACCEPTED.
Mesin direset.
```

Masukan uang sesuai harga minuman

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 5000
ON: Minuman A, Minuman B
Saldo saat ini: Rp5000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 2000
ON: Minuman A, Minuman B, Minuman C
Saldo saat ini: Rp7000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): C
Lintasan DFA: S0 → S5000 → S7000
Minuman C dapat dibeli. Status: ACCEPTED.
Mengembalikan kembalian: Rp1000
Mesin direset.
```

Masukan uang lebih dari harga minuman

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 100
Nominal tidak diterima! Harap masukkan pecahan 1000, 2000, 5000 atau 10000
```

Masukan uang tidak sesuai nominal yang diterima

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): H
Input tidak valid, masukkan nominal angka atau pilihan minuman!
```

Masukan tidak sesuai nominal yang diterima, maupun jenis minuman yang tersedia

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 5000
ON: Minuman A, Minuman B
Saldo saat ini: Rp5000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 5000
ON: Minuman A, Minuman B, Minuman C
Saldo saat ini: Rp10000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 5000
Batas maksimal uang yang dimasukkan adalah Rp10.000!
```

Masukan uang melebihi batas maksimal Rp10.000,00

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): 1000
Saldo saat ini: Rp1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C, 0 untuk keluar): A
Lintasan DFA: S0 → S1000 → REJECT
Saldo tidak cukup. Status: REJECTED
```

Masukan uang kurang dari harga minuman

5. Detail kontribusi masing-masing anggota

Tugas	Aini	Ceiba
Membuat file konfigurasi	✔	✔
Membuat tabel transisi	✔	
Membuat state diagram	✔	✔
Membuat kode program (vending_machine.py)		✔
Membuat kode program dengan fitur pengembalian (vending_machine_withRefund.py)	✔	
Membuat dokumen README.md		✔
Menyusun laporan	✔	✔