LAPORAN PROGRAM VENDING MACHINE BAHASA DAN OTOMATA

Dosen Pembimbing:
Rifki Afina Putri, S.T., M.S.

Anggota Kelompok
Zaky Alraiz Kadarisman (23/516033/PA/22047)
Khairumayzal Dwiksanendra (23/513946/PA/21969)

I. Deskripsi Permasalahan

Mesin penjual otomatis (*vending machine*) digunakan untuk mempermudah transaksi pembelian minuman tanpa perlu interaksi langsung dengan penjual. Namun, sistem ini memerlukan mekanisme yang memastikan bahwa mesin dapat menerima uang dalam nominal tertentu, menghitung total uang yang telah dimasukkan, dan menentukan apakah pengguna dapat membeli produk yang diinginkan.

Permasalahan utama yang dihadapi adalah bagaimana merancang dan mengimplementasikan sistem yang dapat mensimulasikan perilaku *vending machine* menggunakan konsep *Deterministic Finite Automaton* (DFA). Mesin harus mampu memproses transaksi dengan berbagai kondisi, seperti:

1. **Penerimaan Nominal Uang**
   Mesin hanya menerima pecahan tertentu, yaitu Rp1.000, Rp2.000, Rp5.000, dan Rp10.000. Oleh karena itu, sistem harus dapat mengenali dan memvalidasi input uang yang dimasukkan.

2. **Penentuan Keputusan Pembelian**
   Setelah sejumlah uang dimasukkan, pengguna dapat memilih produk yang ingin dibeli. Sistem harus menentukan apakah jumlah uang yang dimasukkan mencukupi harga produk dan memberikan respon yang sesuai.

3. **Penolakan Transaksi**
   Jika jumlah uang yang dimasukkan kurang dari harga produk yang dipilih, mesin harus menolak transaksi dan memberikan informasi bahwa uang tidak cukup.

4. **Mekanisme Pengembalian Uang (Opsional)**
   Jika jumlah uang yang dimasukkan lebih dari harga produk, mesin bisa saja menolak transaksi (jika fitur kembalian tidak ada) atau mengembalikan sisa uang kepada pengguna (jika fitur kembalian diterapkan).

5. **Model DFA untuk Simulasi Mesin**
   Sistem ini diimplementasikan menggunakan DFA, yang memiliki sejumlah state yang merepresentasikan jumlah uang yang telah dimasukkan. Transisi antar-state terjadi berdasarkan input uang atau pilihan produk. State *accepting* menandakan bahwa pengguna dapat membeli minuman sesuai dengan harga yang telah ditentukan.

Permasalahan ini diselesaikan dengan merancang DFA yang menangani setiap kemungkinan transaksi, mengimplementasikannya dalam kode program, dan memastikan validasi input serta keluaran berjalan sesuai dengan spesifikasi sistem yang telah ditetapkan.

II. Notasi DFA

- **States** :

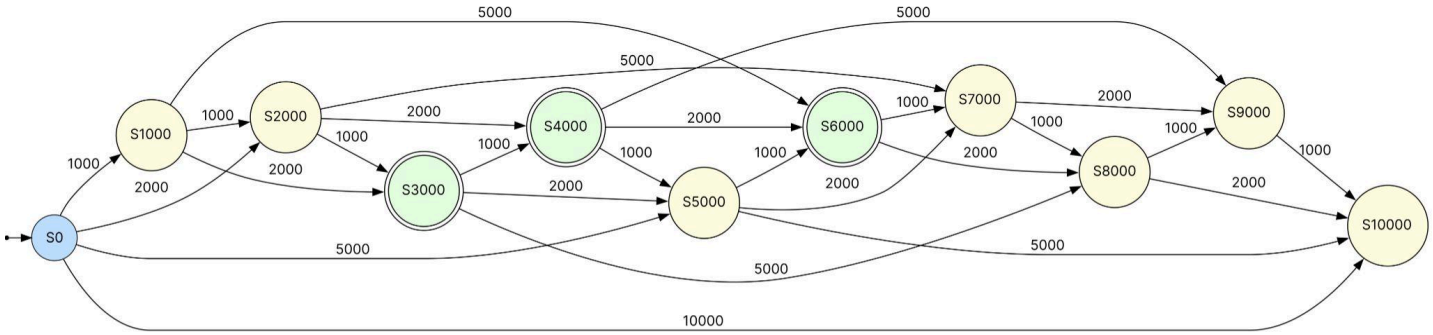
  Q = {S0, S1000, S2000, S3000, S4000, S5000, S6000, S7000, S8000, S9000, S10000}

- **Alphabet :** 1000, 2000, 5000, 10000, A, B, C
- **Start State** : S0
- **Accepting State :** S3000, S4000, S6000
- **Transition Table**

Current State	Input	Next State
S0	1000	S1000
S0	2000	S2000
S0	5000	S5000
S0	10000	S10000
S1000	1000	S2000
S1000	3000	S3000
S1000	6000	S6000
S2000	3000	S3000
S2000	4000	S4000
S2000	7000	S7000
S3000	1000	S4000
S3000	2000	S5000
S3000	5000	S8000
S4000	1000	S5000
S4000	2000	S6000
S4000	5000	S9000
S5000	1000	S6000
S5000	2000	S7000
S5000	5000	S10000
S6000	1000	S7000
S6000	2000	S8000

S7000	1000	S8000
S7000	2000	S9000
S8000	1000	S9000
S8000	2000	S10000
S9000	1000	S10000

- Diagram



III. Impementasi Program

1. Inisialisasi States dan Transisi

```
self.states = {
    'S0': {'1000': 'S1000', '2000': 'S2000', '5000': 'S5000', '10000': 'S10000'},
    'S1000': {'1000': 'S2000', '2000': 'S3000', '5000': 'S6000', '10000': 'S11000'},
    'S2000': {'1000': 'S3000', '2000': 'S4000', '5000': 'S7000', '10000': 'S12000'},
    'S3000': {'1000': 'S4000', '2000': 'S5000', '5000': 'S8000', '10000': 'S13000'},
    'S4000': {'1000': 'S5000', '2000': 'S6000', '5000': 'S9000', '10000': 'S14000'},
    'S5000': {'1000': 'S6000', '2000': 'S7000', '5000': 'S10000', '10000': 'S15000'},
    'S6000': {'1000': 'S7000', '2000': 'S8000', '5000': 'S11000', '10000': 'S16000'},
    'S7000': {'1000': 'S8000', '2000': 'S9000', '5000': 'S12000', '10000': 'S17000'},
    'S8000': {'1000': 'S9000', '2000': 'S10000', '5000': 'S13000', '10000': 'S18000'},
    'S9000': {'1000': 'S10000', '2000': 'S11000', '5000': 'S14000', '10000': 'S19000'},
    'S10000': {'1000': 'S11000', '2000': 'S12000', '5000': 'S15000', '10000': 'S20000'}
}
```

Bagian ini mendefinisikan state transition yang mengatur perpindahan dari satu state ke state lain berdasarkan jumlah yang dimasukkan.

Contohnya :

- Dari S0, jika pengguna melakukan input 1000, state akan berpindah ke S1000
- Dari S1000, jika pengguna memasukkan 2000, state akan berpindah ke S300

2. **Inisialisasi Variabel Lain**

```
self.current_state = 'S0'
self.accepting_states = {'S3000', 'S4000', 'S6000'}
self.path = ['S0']
self.total_money = 0
self.drink_prices = {'A': 3000, 'B': 4000, 'C': 6000}
```

- self.current _state : Menyimpan state awal (S0)
- self.accepting _states : menyimpan state yang memungkinkan pembelian minuman (S3000, S4000, S6000)
- self.path : menyimpan jalur pergerakan DFA
- Self.total_money : menyimpan total uang yang dimasukkan oleh pengguna
- Self.drink_prices : menimpan harga dari masing-masing minuman

3. **Metode reset()**

```
def reset(self):
    self.current_state = 'S0'
    self.path = ['S0']
    self.total_money = 0
```

Bagian kode ini digunakan untuk mereset vending machine setelah pembelian berhasil atau gagal

4. **Metode process_input(input_value)**

```
def process_input(self, input_value):
    if input_value in {'1000', '2000', '5000', '10000'}:
        money = int(input_value)
        if self.total_money + money > 10000:
            print("Error: Jumlah uang melebihi Rp10.000")
            return False
```

Bagian kode ini berfungsi untuk memeriksa apakah input adalah pecahan unag yang valid. Jika total uang melebihi 10.000, maka transaksi akan ditolak

```
self.total_money += money
if self.current_state in self.states and input_value in self.states[self.current_state]:
    self.current_state = self.states[self.current_state][input_value]
    self.path.append(self.current_state)
    return True
```

Bagian kode di atass berfungsi untuk menambah jumlah uang ke self.total_money dan memeriksa apakah ada transisi yang sesuai dalam DFA, lalu memperbarui current_state

```
elif input_value in {'A', 'B', 'C'}:
    return self.process_purchase(input_value)
else:
    print("Input tidak valid")
    return False
```

Bagian kode ini akan memproses pilihan minuman (A, B, C) berdasarkan input user melalui method process_purchase()

5. **Metode process_purchase(drink)**

```
def process_purchase(self, drink):
    price = self.drink_prices[drink]
    if self.total_money >= price:
        change = self.total_money - price
        print(f"Lintasan DFA: {' → '.join(self.path)}")
        print(f"Minuman {drink} dapat dibeli. Status: ACCEPTED.")
        if change > 0:
            print(f"Kembalian: {change}")
        self.reset()
        return True
```

Bagian kode ini akan memeriksa apakah uang cukup untuk membeli minuman. Jika cukup, akan mencetak lintasan DFA dan memberikan kembalian jika ada.

6. **Metode check_drink_availability()**

```
def check_drink_availability(self):
    available = []
    if self.total_money >= 3000:
        available.append('A')
    if self.total_money >= 4000:
        available.append('B')
    if self.total_money >= 6000:
        available.append('C')

    if available:
        print(f"ON: Minuman {', '.join(available)}")
    return available
```

Bagian kode ini akan menampilkan minuman yang bisa dibeli berdasarkan jumlah total uang yang sudah dimasukan oleh user.

7. **Fungsi main()**

```python
def main():
    vending_machine = VendingMachineDFA()
    valid_inputs = ['1000', '2000', '5000', '10000', 'A', 'B', 'C']

    while True:
        user_input = input(f"Masukkan uang atau beli minuman ({', '.join(valid_inputs)}): ").strip().upper()

        if user_input == 'EXIT':
            break

        if user_input not in valid_inputs:
            print("Input tidak valid. Masukkan pecahan uang atau pilihan minuman.")
            continue

        if vending_machine.process_input(user_input):
            if user_input in {'1000', '2000', '5000', '10000'}:
                vending_machine.check_drink_availability()
```

Bagian kode ini berfungsi sebagai interface utama yang berinteraksi dengan user dan mengontrol jalannya vending machine berdasarkan input yang diberikan. Berikut alur utama dalam fungsi ini :

- Program membuat objek VendingMachineDFA untuk menangani transaksi
- Menentukan daftar input yang valid (uang : 1000, 2000, 5000, 10000 dan minuman : A, B, C)
- Program akan meminta pengguna memasukan uang atau memiliki minuman. Jika input adalah “EXIT”, maka program akan berhenti. Jika untpu tidak valid, akan menampilkan pesan error dan user diminta ulangin input
- Jika user melakukan input berupa uang, sistem akan memperbarui jumlah uang dan menampilkan minuman yang bisa dibeli.
- Jika input adalah minuman, sistem mengecek apakah uang cukup untuk pembelian
- Jika jumlah uang melebihi 10.000, transaksi akan ditolak

IV. Input dan Output

1. **Pembelian dengan uang pas**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 2000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
ON: Minuman A
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): A
Lintasan DFA: S0 → S2000 → S3000
Minuman A dapat dibeli. Status: ACCEPTED.
```

2. **Pembelian dengan uang lebih dan program memberikan kembalian**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, B
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): A
Lintasan DFA: S0 → S5000
Minuman A dapat dibeli. Status: ACCEPTED.
Kembalian: 2000
```

3. **Uang kurang untuk membeli minuman**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): A
Lintasan DFA: S0 → S1000
Uang tidak cukup. Status: REJECTED.
```

4. **User memasukan uang lebih dari batas maksimun (10.000)**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, B
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
ON: Minuman A, B, C
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 2000
Error: Jumlah uang melebihi Rp10.000
```

5. **User melakukan input yang tidak valid**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): cihuy
Input tidak valid. Masukkan pecahan uang atau pilihan minuman.
```

6. **Pembatalan transaksi dengan EXIT**

```
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 10000
ON: Minuman A, B, C
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): EXIT
PS C:\Users\Ical\Documents\Main File\UGM\Tugas Kuliah\kodang\Test>
```

V. Kontribusi Anggota Tim

Khairumayzal Dwiksanendra :

- Mengerjakan dan mengkoreksi beberapa fungsi pada program mengacu pada instruksi pengerjaan
- Mengerjakan sebagian besar laporan

Zaky Alraiz :

- Mengerjakan Sebagian besar program
- Mengupdate beberapa bagian dalam laporan sesuai dengan bagian program yang telah dikoreksi