Name: ANAS JAWAID

ID: CSC-20F-028

SECTION: 4F

COURSE: THEORY OF AUTAMATA

INSTRUCTOR: MUSTAFA ALI BOOMBAT

ASSIGNMENT # 2

Q1. Build an FA accepting the Language L of Strings, defined over Σ = {a, b}, beginning with and ending in same letters;

Construction of the Finite Automaton (FA)

The FA will have the following states:

1. q0q_0q0: Start state (also an accepting state if the string is empty).
2. q1q_1q1: A state indicating the string started with 'a'.
3. q2q_2q2: A state indicating the string started with 'b'.
4. q3q_3q3: An accepting state indicating a valid string that started and ended with 'a'.
5. q4q_4q4: An accepting state indicating a valid string that started and ended with 'b'.

Transition Rules

1. From q0q_0q0 (start state):
    - On input 'a', move to q1q_1q1.
    - On input 'b', move to q2q_2q2.
2. From q1q_1q1 (string started with 'a'):
    - On input 'a', stay in q1q_1q1 (continue tracking a string that starts with 'a').
    - On input 'b', move to q3q_3q3 (indicates switching, but we track the possibility of ending with 'a').
3. From q2q_2q2 (string started with 'b'):
    - On input 'a', move to q4q_4q4 (indicates switching, but we track the possibility of ending with 'b').
    - On input 'b', stay in q2q_2q2 (continue tracking a string that starts with 'b').
4. From q3q_3q3 (potential state for accepting a string starting and ending with 'a'):
    - On input 'a', move back to q1q_1q1.
    - On input 'b', stay in q3q_3q3.

5. From q4q_4q4 (potential state for accepting a string starting and ending with 'b'):
    - On input 'a', stay in q4q_4q4.
    - On input 'b', move back to q2q_2q2.

Accepting States

1. q1q_1q1 and q3q_3q3: Accepts strings starting and ending with 'a'.
2. q2q_2q2 and q4q_4q4: Accepts strings starting and ending with 'b'.

a a

----> (q1) ----------> (q_accept_a)

| | b

| |

b| |

| v

(q0) (q2) ----------> (q_accept_b)

| | a

----> (q1) ----------> (q_accept_a)

Formal Definition of the FA

1. **States**: Q={q0,q1,q2,qaccepta,qacceptb}Q = \{q0, q1, q2, q_accept_a, q_accept_b\}Q={q0,q1,q2,qaccepta,qacceptb}
2. **Alphabet**: Σ={a,b}\Sigma = \{a, b\}Σ={a,b}
3. **Start state**: q0q0q0
4. **Accept states**: F={qaccepta,qacceptb}F = \{q_accept_a, q_accept_b\}F={qaccepta,qacceptb}
5. **Transition function**:

- δ(q0,a)=q1\delta(q0, a) = q1δ(q0,a)=q1
- δ(q0,b)=q2\delta(q0, b) = q2δ(q0,b)=q2
- δ(q1,a)=q1\delta(q1, a) = q1δ(q1,a)=q1
- δ(q1,b)=q1\delta(q1, b) = q1δ(q1,b)=q1
- δ(q2,a)=q2\delta(q2, a) = q2δ(q2,a)=q2
- δ(q2,b)=q2\delta(q2, b) = q2δ(q2,b)=q2

Q2. Using the technique discussed by Martin, build an FA accepting the following language L = {w belongs to {a,b}*: length(w) >= 2 and second letter of w, from right is a}.

1. q0: Start state.
2. q1: State reached after reading the first character.
3. q2: State reached after reading a string with the second-to-last character as 'a'.
4. q3: State for tracking a string with the second-to-last character as 'b'.
5. q4: Accepting state where the condition is met (second-to-last character is 'a').

1. From q0 (start state):

- On input 'a', move to q1.
- On input 'b', move to q1.

2. From q1 (after reading the first character):

- On input 'a', move to q2 (indicating the current character is 'a').
- On input 'b', move to q3 (indicating the current character is 'b').

3. From q2q_2q2 (tracking the second-to-last character as 'a'):

- On input 'a' or 'b', move to q4 (final state; the second-to-last character remains 'a').

4. **From q3q_3q3 (tracking the second-to-last character as 'b'):**

- On input 'a', move to q2 (updating to indicate the second-to-last character is now 'a').
- On input 'b', stay in q3 (maintaining the second-to-last character as 'b').

5. **From q4q_4q4 (accepting state):**

- On input 'a' or 'b', stay in q4q_4q4 (remaining in the accepting condition).

Accepting State:

- q4: The automaton accepts when it reaches q4, indicating that the string's second-to-last character is 'a'.

transition Table:

Current State	Input 'a'	Input 'b'
qo	q1	q1
q1	q2	q3
q2	q4	q4
q3	q2	q3
q4	q4	q4

Diagram Representation

The state diagram would have transitions as follows:

q0→q1 on 'a' or 'b'.

q1→q2 on 'a', q1→q3q_1 \rightarrow q_3q1→q3 on 'b'.

- q2→q4 on 'a' or 'b'.
- q3→q2q_3 \rightarrow q_2q3→q2 on 'a', q3→q3q_3 \rightarrow q_3q3→q3 on 'b'.

- q4→q4q_4 \rightarrow q_4q4→q4 on 'a' or 'b'.

Q3. Using the technique discussed by Martin, build an FA accepting the following language L = {w belongs to {a,b}*: w neither ends in ab nor ba}.

1. q0: Start state and also an accepting state for strings shorter than 2 characters.
2. q1: State where the last character read was 'a'.
3. q2: State where the last character read was 'b'.
4. qaa: State indicating the last two characters are "aa".
5. qbb: State indicating the last two characters are "bb".
6. qab: State indicating the last two characters are "ab" (rejecting condition).
7. qba: State indicating the last two characters are "ba" (rejecting condition).

1. **From q0q_0q0**:

- On input 'a', move to q1.
- On input 'b', move to q2.

2 . From q1q_1q1 (last character is 'a'):

- On input 'a', move to qaa (sequence ends in "aa").
- On input 'b', move to qab (sequence ends in "ab").

3. **From q2q_2q2** (last character is 'b'):

- On input 'a', move to qba (sequence ends in "ba").
- On input 'b', move to qbb (sequence ends in "bb")

4. **From qaaq_{aa}qaa** (last two characters are "aa"):

- On input 'a', stay in qaa.
- On input 'b', move to qab.

5. **From qbbq_{bb}qbb** (last two characters are "bb"):

- On input 'a', move to qba.
- On input 'b', stay in qbb.

6. **From qabq_{ab}qab** (last two characters are "ab"):

- On input 'a', move to qaa.
- On input 'b', move to qbb.

7. **From qbaq_{ba}qba** (last two characters are "ba"):

- On input 'a', move to qaa.
- On input 'b', move to qbb.

Transition Table:

Current State	***Input 'a'***	***Input 'b'***
Qo	q1	q2
Q1	qaa	qab
Q2	qba	qbb
Qaa	qaa	qab
Qbb	qba	qbb

Qab	qaa	qbb
Qba	qaa	qbb

Q4. Build a TG accepting the language of strings, defined over Σ={a, b}, ending in b.

1. From q0:

- On input 'a', stay in q0 or move to q2 (as the string has not ended with 'b' yet).
- On input 'b', move to q1 (indicating the string ends in 'b').

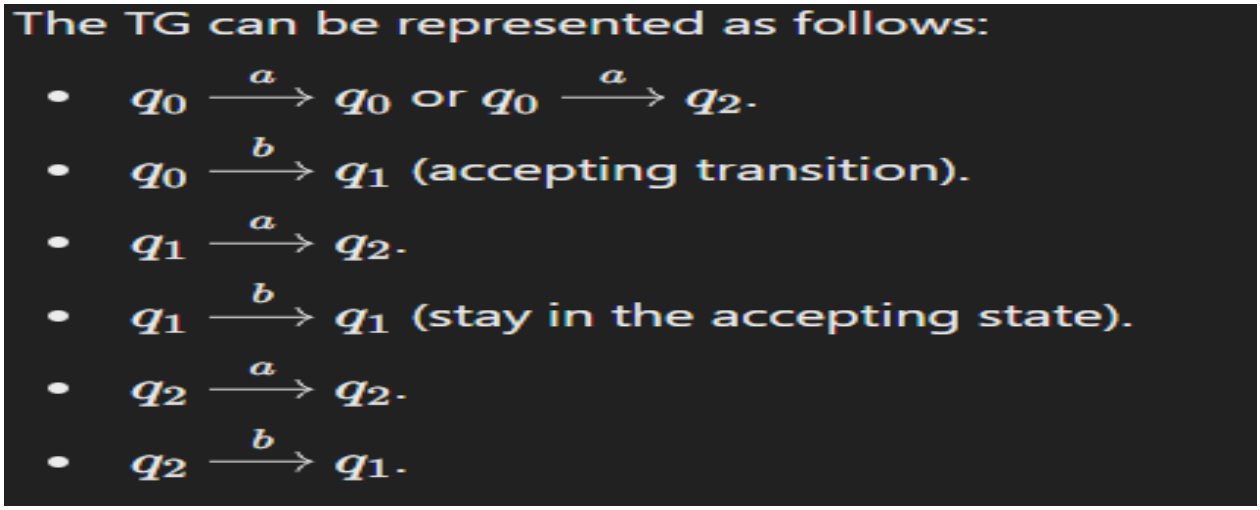
2. From q1q_1q1:

- On input 'a', move to q2 (indicating that the string no longer ends in 'b').
- On input 'b', stay in q1 (string continues to end in 'b').

3. From q2q_2q2:

- On input 'a', stay in q2q_2q2 (string still does not end in 'b').
- On input 'b', move to q1q_1q1 (indicating the string now ends in 'b').

The TG can be represented as follows:



1. The TG starts at q0. It processes input characters and tracks whether the string currently ends in 'b'.
2. If the last character read is 'b', the automaton stays in q1, which is the accepting state.
3. If an 'a' is read, the automaton moves to q2, indicating that the string does not currently end in 'b'.
4. q1 is the accepting state because it represents the condition where the last character of the string is 'b'.

- ## Transition Table

Current State	Input 'a'	Input 'b'
Q0	q0/q2	q1
Q1	q2	q1
Q2	q2	q1

Example Strings

- **"ab"**: Accepted (ends in 'b').
- **"b"**: Accepted (ends in 'b').
- **"ba"**: Not accepted (ends in 'a').
- **"a"**: Not accepted (ends in 'a').
- **"bab"**: Accepted (ends in 'b').

This TG correctly accepts all strings defined over Σ={a,b}\Sigma = \{a, b\}Σ={a,b} that end in 'b'.

Q5. Build a TG accepting the language L of strings, defined over Σ={a, b}, beginning with and ending in the same letters.

1. **q0**: Start state where no input has been read yet.
2. **qstart_a**: State reached when the string begins with 'a'.
3. **qstart_b**: State reached when the string begins with 'b'.
4. **qaccept_a**: Accepting state for strings that begin and end with 'a'.
5. **qaccept_b**: Accepting state for strings that begin and end with 'b'.
6. **qreject**: State for strings that do not match the starting condition at the end (not used in transitions but conceptual for rejecting).

Transitions

1.From q0:

a. On input 'a', move to qstart_aq_{start_a}qstart_a.
b. On input 'b', move to qstart_bq_{start_b}qstart_b.

2. **From qstart_a**:

- On input 'a', stay in qstart_aq_{start_a}qstart_a or move to qaccept_aq_{accept_a}qaccept_a when reaching the end.
- On input 'b', stay in qstart_aq_{start_a}qstart_a (still processing the string).

3. **From qstart_b**:

- On input 'b', stay in qstart_bq_{start_b}qstart_b or move to qaccept_bq_{accept_b}qaccept_b when reaching the end.
- On input 'a', stay in qstart_bq_{start_b}qstart_b (still processing the string).

4. **From qaccept_b**:

- These states are accepting and indicate that the last character matches the first.
- **Transition Table**

Current State	Input 'a'	Input 'b'
Q0	qstart_a	qstart_b
Qstart_a	qstart_a/qaccept_b	qstart_a
Qstart_b	qstart_b	qstart_b/qaccept_b
Qaccept_a	qaacept_a	qaccept_a
Qaccept_b	qaccept_b	qaccept_b

Example Strings

- **"a"**: Accepted (begins and ends with 'a').
- **"b"**: Accepted (begins and ends with 'b').
- **"ab"**: Not accepted (begins with 'a' and ends with 'b').
- **"ba"**: Not accepted (begins with 'b' and ends with 'a').
- **"aba"**: Accepted (begins and ends with 'a').
- **"bab"**: Accepted (begins and ends with 'b').

This TG correctly accepts all strings that begin and end with the same letter over Σ={a,b}\Sigma = \{a, b\}Σ={a,b}.