Toc practicals:-

Name :-Tushar

Roll no :- 16085

Qno1.

```
#include <iostream>
#include <string>
using namespace std;

// Forward declaration of states
void State0(const string &w, int i);
void State1(const string &w, int i);
void State2(const string &w, int i);
void State3(const string &w, int i);

// State 0: Start state
void State0(const string &w, int i) {
    cout << "State 0" << endl;
    if (i == w.size()) { // Use .size() to check the length of the string
        cout << "String is rejected." << endl; // Didn't reach final state
        return;
    }
    if (w[i] == '1')
        State1(w, i + 1);
    else
        State0(w, i + 1); // Stay in q0 for input 0
}

// State 1: One consecutive '1' encountered
void State1(const string &w, int i) {
    cout << "State 1" << endl;
    if (i == w.size()) { // Use .size() to check the length of the string
        cout << "String is rejected." << endl; // Didn't reach final state
        return;
    }
    if (w[i] == '1')
        State2(w, i + 1);
    else
```

```
        State0(w, i + 1); // Reset to q0 on input 0
}

// State 2: Two consecutive '1's encountered
void State2(const string &w, int i) {
    cout << "State 2" << endl;
    if (i == w.size()) { // Use .size() to check the length of the string
        cout << "String is rejected." << endl; // Didn't reach final state
        return;
    }
    if (w[i] == '1')
        State3(w, i + 1);
    else
        State0(w, i + 1); // Reset to q0 on input 0
}

// State 3: Final state, three consecutive '1's
void State3(const string &w, int i) {
    cout << "State 3" << endl;
    if (i == w.size()) { // Use .size() to check the length of the string
        cout << "String is accepted." << endl; // Reached final state
        return;
    }
    State3(w, i + 1); // Stay in final state for any input
}

// Main function
int main() {
    string w;
    cout << "Enter a binary string: ";
    cin >> w;
    State0(w, 0); // Start with State0
    return 0;
}
```

Qno2.

```
#include <iostream>
#include <string>
using namespace std;

// Forward declarations for state functions
void State0(const string &w, int i);
void State1(const string &w, int i);
```

```
void State2(const string &w, int i);
void State3(const string &w, int i);
void State4(const string &w, int i); // Trap state

// State 0: Start state
void State0(const string &w, int i) {
    cout << "State 0" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == '1')
        State1(w, i + 1);
    else
        State0(w, i + 1);
}

// State 1: One '1' encountered
void State1(const string &w, int i) {
    cout << "State 1" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == '1')
        State2(w, i + 1);
    else
        State1(w, i + 1);
}

// State 2: Two '1's encountered (final state for two '1's)
void State2(const string &w, int i) {
    cout << "State 2" << endl;
    if (i == w.size()) {
        cout << "String is accepted. (Exactly two '1's)" << endl;
        return;
    }
    if (w[i] == '1')
        State3(w, i + 1);
    else
        State2(w, i + 1);
}

// State 3: Three '1's encountered (final state for three '1's)
```

```
void State3(const string &w, int i) {
    cout << "State 3" << endl;
    if (i == w.size()) {
        cout << "String is accepted. (Exactly three '1's)" << endl;
        return;
    }
    if (w[i] == '1')
        State4(w, i + 1);
    else
        State3(w, i + 1);
}

// State 4: More than three '1's encountered (trap state)
void State4(const string &w, int i) {
    cout << "State 4 (Trap State)" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    State4(w, i + 1); // Stay in trap state for any input
}

// Main function
int main() {
    string w;
    cout << "Enter a binary string: ";
    cin >> w;
    State0(w, 0); // Start with State0
    return 0;
}
```

Qno3.

```
 #include <iostream>
#include <string>
using namespace std;

// Forward declarations for state functions
void State0(const string &w, int i, char first, char second);
void State1(const string &w, int i, char first, char second);
void State2(const string &w, int i, char first, char second);
void State3(const string &w, int i, char first, char second);
void State4(const string &w, int i, char first, char second);
void StateReject(const string &w);
```

```
// State 0: Start state
void State0(const string &w, int i, char first, char second) {
    cout << "State 0" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State1(w, i + 1, w[i], '\0');
    else
        StateReject(w);
}

// State 1: Read the second character
void State1(const string &w, int i, char first, char second) {
    cout << "State 1" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State2(w, i + 1, first, w[i]);
    else
        StateReject(w);
}

// State 2: Process intermediate characters
void State2(const string &w, int i, char first, char second) {
    cout << "State 2" << endl;
    if (i >= w.size() - 2) { // Only last two characters are left
        State3(w, i, first, second);
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State2(w, i + 1, first, second);
    else
        StateReject(w);
}

// State 3: Read the third-to-last character
void State3(const string &w, int i, char first, char second) {
    cout << "State 3" << endl;
    if (i >= w.size()) {
```

```
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == first)
        State4(w, i + 1, first, second);
    else
        StateReject(w);
}

// State 4: Read the fourth-to-last character and verify
void State4(const string &w, int i, char first, char second) {
    cout << "State 4" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == second) {
        cout << "String is accepted." << endl;
        return;
    }
    StateReject(w);
}

// Rejection state
void StateReject(const string &w) {
    cout << "Rejection State" << endl;
    cout << "String is rejected." << endl;
}

// Main function
int main() {
    string w;
    cout << "Enter a string over {a, b}: ";
    cin >> w;
    if (w.size() < 4) {
        cout << "String is rejected. (Too short)" << endl;
    } else {
        State0(w, 0, '\0', '\0'); // Start at State 0
    }
    return 0;
}
```

Qno4.

```
#include <iostream>
#include <string>
using namespace std;

// Forward declarations for state functions
void State0(const string &w, int i);
void State1(const string &w, int i);
void State2(const string &w, int i);
void #include <iostream>
#include <string>
using namespace std;

// Forward declarations for state functions
void State0(const string &w, int i, char first, char second);
void State1(const string &w, int i, char first, char second);
void State2(const string &w, int i, char first, char second);
void State3(const string &w, int i, char first, char second);
void State4(const string &w, int i, char first, char second);
void StateReject(const string &w);

// State 0: Start state
void State0(const string &w, int i, char first, char second) {
    cout << "State 0" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State1(w, i + 1, w[i], '\0');
    else
        StateReject(w);
}

// State 1: Read the second character
void State1(const string &w, int i, char first, char second) {
    cout << "State 1" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State2(w, i + 1, first, w[i]);
    else
        StateReject(w);
```

```
}

// State 2: Process intermediate characters
void State2(const string &w, int i, char first, char second) {
    cout << "State 2" << endl;
    if (i >= w.size() - 2) { // Only last two characters are left
        State3(w, i, first, second);
        return;
    }
    if (w[i] == 'a' || w[i] == 'b')
        State2(w, i + 1, first, second);
    else
        StateReject(w);
}

// State 3: Read the third-to-last character
void State3(const string &w, int i, char first, char second) {
    cout << "State 3" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == first)
        State4(w, i + 1, first, second);
    else
        StateReject(w);
}

// State 4: Read the fourth-to-last character and verify
void State4(const string &w, int i, char first, char second) {
    cout << "State 4" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == second) {
        cout << "String is accepted." << endl;
        return;
    }
    StateReject(w);
}

// Rejection state
void StateReject(const string &w) {
```

```
    cout << "Rejection State" << endl;
    cout << "String is rejected." << endl;
}

// Main function
int main() {
    string w;
    cout << "Enter a string over {a, b}: ";
    cin >> w;
    if (w.size() < 4) {
        cout << "String is rejected. (Too short)" << endl;
    } else {
        State0(w, 0, '\0', '\0'); // Start at State 0
    }
    return 0;
}
(const string &w);

// State 0: Start state
void State0(const string &w, int i) {
    cout << "State 0" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a')
        State1(w, i + 1);
    else
        StateReject(w);
}

// State 1: After reading the first 'a'
void State1(const string &w, int i) {
    cout << "State 1" << endl;
    if (i >= w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'b' && i == w.size() - 1) {
        // If 'b' is the last character, transition to State 2
        State2(w, i + 1);
    } else if (w[i] == 'a' || w[i] == 'b') {
        // Loop in State 1 for middle characters
        State1(w, i + 1);
```

```
    } else {
        StateReject(w);
    }
}

// State 2: Final state after reading 'b'
void State2(const string &w, int i) {
    cout << "State 2" << endl;
    if (i == w.size()) {
        cout << "String is accepted." << endl;
        return;
    }
    StateReject(w); // No valid transitions from this state
}

// Rejection state
void StateReject(const string &w) {
    cout << "Rejection State" << endl;
    cout << "String is rejected." << endl;
}

// Main function
int main() {
    string w;
    cout << "Enter a string over {a, b}: ";
    cin >> w;
    if (w.size() < 2) {
        cout << "String is rejected. (Too short)" << endl;
    } else {
        State0(w, 0); // Start at State 0
    }
    return 0;
}
```

Qno5.

```
#include <iostream>
#include <string>
using namespace std;

// Function prototypes for states
void State00(const string &w, int i);
void State01(const string &w, int i);
void State10(const string &w, int i);
```

```
void State11(const string &w, int i);

// State 00: Even 'a's and even 'b's (Accepting state)
void State00(const string &w, int i) {
    cout << "State 00 (Even 'a', Even 'b')" << endl;
    if (i == w.size()) {
        cout << "String is accepted." << endl;
        return;
    }
    if (w[i] == 'a') State10(w, i + 1);
    else if (w[i] == 'b') State01(w, i + 1);
    else cout << "Invalid input. String rejected." << endl;
}

// State 01: Even 'a's and odd 'b's
void State01(const string &w, int i) {
    cout << "State 01 (Even 'a', Odd 'b')" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a') State11(w, i + 1);
    else if (w[i] == 'b') State00(w, i + 1);
    else cout << "Invalid input. String rejected." << endl;
}

// State 10: Odd 'a's and even 'b's
void State10(const string &w, int i) {
    cout << "State 10 (Odd 'a', Even 'b')" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
    }
    if (w[i] == 'a') State00(w, i + 1);
    else if (w[i] == 'b') State11(w, i + 1);
    else cout << "Invalid input. String rejected." << endl;
}

// State 11: Odd 'a's and odd 'b's
void State11(const string &w, int i) {
    cout << "State 11 (Odd 'a', Odd 'b')" << endl;
    if (i == w.size()) {
        cout << "String is rejected." << endl;
        return;
```

```
    }
    if (w[i] == 'a') State01(w, i + 1);
    else if (w[i] == 'b') State10(w, i + 1);
    else cout << "Invalid input. String rejected." << endl;
}

// Main function
int main() {
    string w;
    cout << "Enter a string over {a, b}: ";
    cin >> w;
    State00(w, 0); // Start at State 00
    return 0;
}
```

Qno6.

```
#include <iostream>
#include <string>
#include <set>
using namespace std;

// Simulating FAs for L1 and L2
bool simulateL1(const string &w) {
    // FA for L1: Strings with an even number of 'a's
    int countA = 0;
    for (char c : w) {
        if (c == 'a') countA++;
    }
    return countA % 2 == 0;
}

bool simulateL2(const string &w) {
    // FA for L2: Strings with an odd number of 'b's
    int countB = 0;
    for (char c : w) {
        if (c == 'b') countB++;
    }
    return countB % 2 != 0;
}

// Simulate Union (L1 U L2)
bool simulateUnion(const string &w) {
    return simulateL1(w) || simulateL2(w);
```

```
}

// Simulate Intersection (L1 ∩ L2)
bool simulateIntersection(const string &w) {
    return simulateL1(w) && simulateL2(w);
}

// Simulate Concatenation (L1 L2)
bool simulateConcatenation(const string &w) {
    for (size_t i = 0; i <= w.size(); i++) {
        string part1 = w.substr(0, i);
        string part2 = w.substr(i);
        if (simulateL1(part1) && simulateL2(part2)) {
            return true;
        }
    }
    return false;
}

// Main function
int main() {
    string w;
    cout << "Enter a string over {a, b}: ";
    cin >> w;

    cout << "Simulating L1 (even number of 'a's): "
         << (simulateL1(w) ? "Accepted" : "Rejected") << endl;

    cout << "Simulating L2 (odd number of 'b's): "
         << (simulateL2(w) ? "Accepted" : "Rejected") << endl;

    cout << "Union (L1 U L2): "
         << (simulateUnion(w) ? "Accepted" : "Rejected") << endl;

    cout << "Intersection (L1 ∩ L2): "
         << (simulateIntersection(w) ? "Accepted" : "Rejected") << endl;

    cout << "Concatenation (L1 L2): "
         << (simulateConcatenation(w) ? "Accepted" : "Rejected") << endl;

    return 0;
}
```