**TOC Practical File**

**Name:** Shivam Kumar

**Course:** B.Sc. (hons.) Computer Science – VB

**Roll No**: 16076

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Practical Question** | **Pg no.** |
| 1. | Finite Automata accepts having 3 consecutive 1’s as substring | 3 |
| 2. | Finite Automata that accept either exactly two 1’s or exactly 3 1’s | 5 |
| 3. | Finite Automata accepts L comprising of all string having first two characters same as the last two | 8 |
| 4. | Finite Automata accepts where L = a (a + b )\*b | 12 |
| 5. | Finite automata accepts EVEN-EVEN over S={a, b}. | 15 |
| 6. | Program to simulate an FA that accepts  Union Intersection and L1L2 concatenation | 17 |
| 7. | PDA for language a^n b^n where n>0 | 20 |
| 8. | PDA for language {w X w^r | w is any string over S = {a,b} | 22 |
| 9. | Turing Machine for a^n b^n c^n where n>0 | 24 |
| 10. | Turing Machine which will increment the given binary number by 1. | 26 |

**Que.1**

Design a Finite Automata (FA) that accepts all strings over S={0, 1} having three consecutive 1's as a substring. Write a program to simulate this FA.

**Code Snippet:**

#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: ";

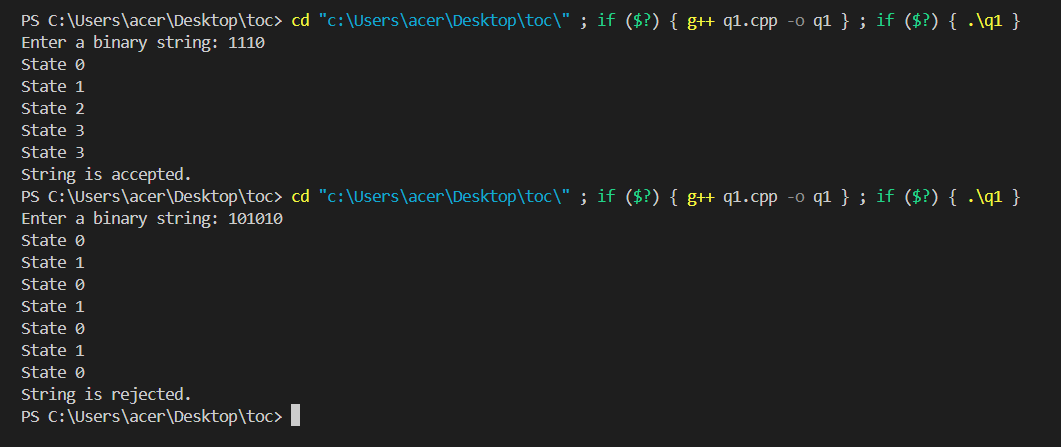
cin >> w;

State0(w, 0); // Start with State0

return 0;

}

**Output:**



**Que.2**

Design a Finite Automata (FA) that accepts all strings over S={0, 1} having either exactly two 1's or exactly three 1's, not more nor less. Write a program to simulate this FA.

**Code Snippet:**

#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: ";

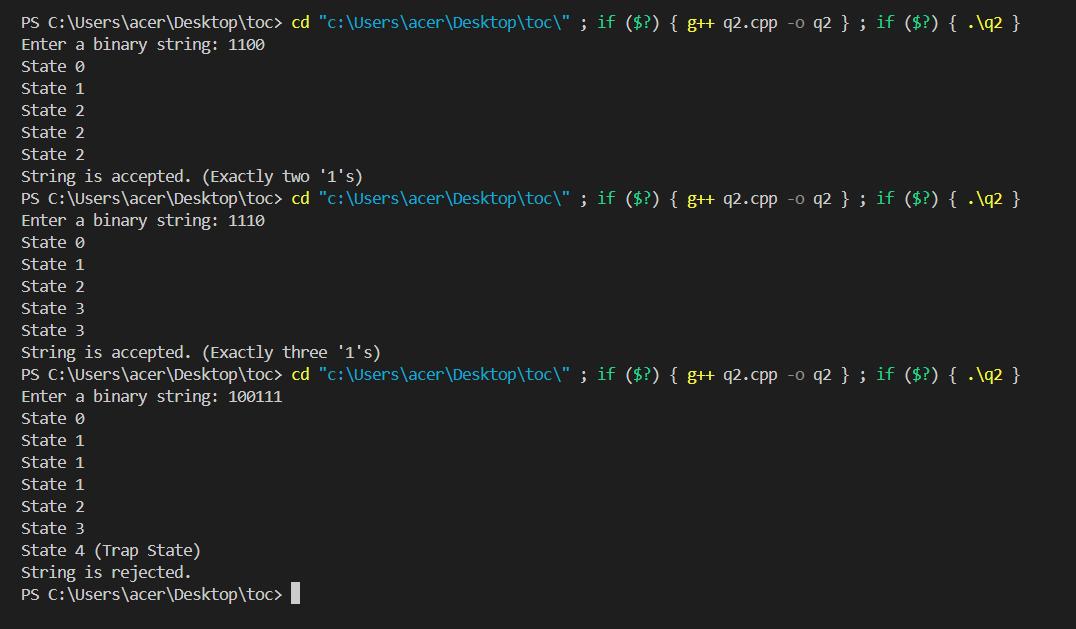
cin >> w;

State0(w, 0); // Start with State0

return 0;

}

**Output:**



**Que.3**

Design a Finite Automata (FA) that accepts language L1, over S={a, b}, comprising of all strings (of length 4 or more) having first two characters same as the last two. Write a program to simulate this FA.

**Code Snippet:**

#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 {

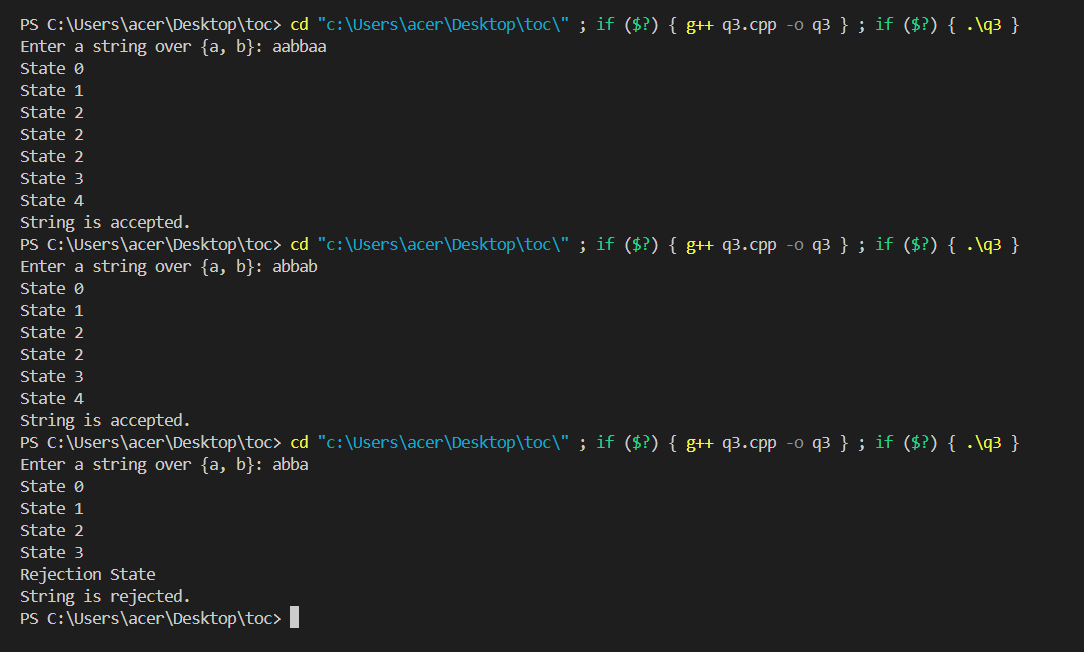
        State0(w, 0, '\0', '\0'); // Start at State 0

    }

    return 0;

}

**Output:**



**Que.4**

Design a Finite Automata (FA) that accepts language L2, over S= {a, b} where L2= a(a+b)\*b. Write a program to simulate this FA.

**Code Snippet:**

#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 StateReject(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 {

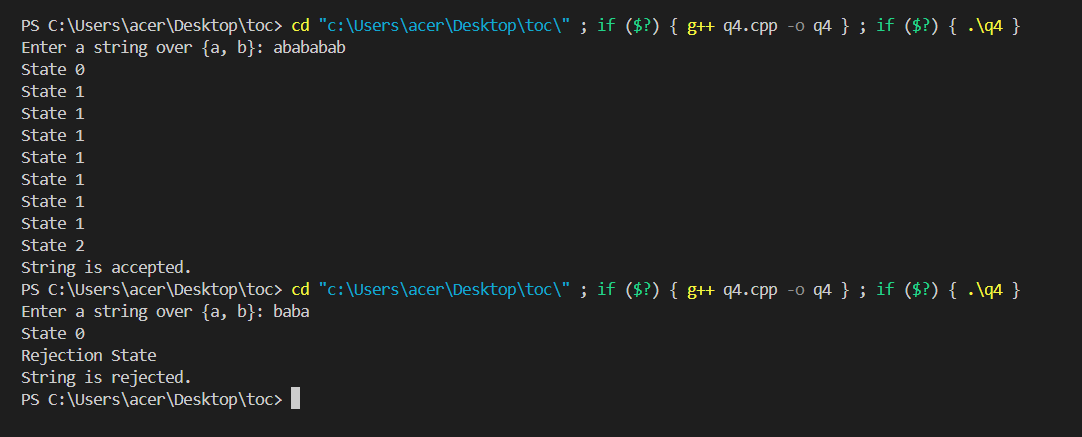
        State0(w, 0); // Start at State 0

    }

    return 0;

}

**Output:**



**Que.5**

Design a Finite Automata (FA) that accepts language EVEN-EVEN over S={a,b}. Write a program to simulate this FA.

**Code Snippet:**

#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}: ";

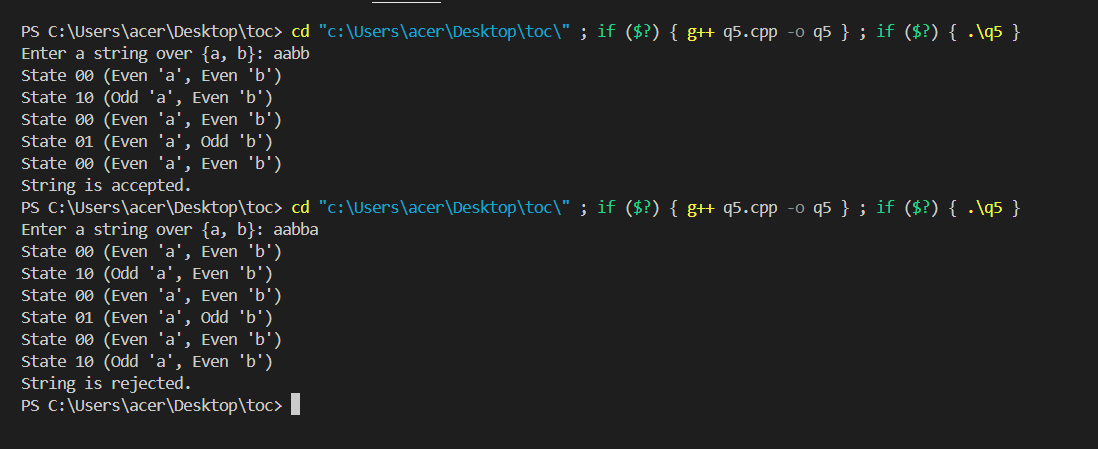
    cin >> w;

    State00(w, 0); // Start at State 00

    return 0;

}

**Output:**



**Que.6**

Write a program to simulate an FA that accepts

a. Union of the languages L1 and L2

b. Intersection of the languages L1 and L2

c. Language L1 L2 (concatenation)

**Code Snippet:**

#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;

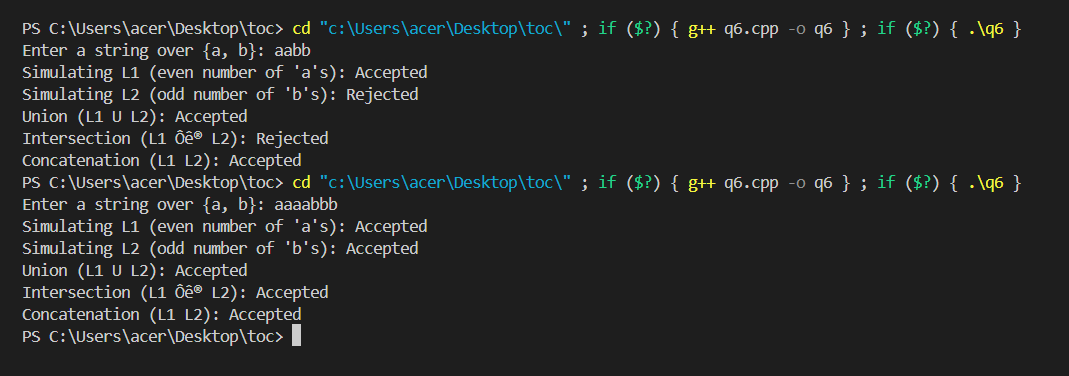
    cout << "Concatenation (L1 L2): "

         << (simulateConcatenation(w) ? "Accepted" : "Rejected") << endl;

    return 0;

}

**Output:**



**Que.7**

Design a PDA and write a program for simulating the machine which accepts the language {a^n b^n where n>0, S= {a, b}}.

**Code Snippet:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Function to simulate PDA

bool simulatePDA(const string &input) {

    stack<char> pdaStack;

    int i = 0;

    // Transition for pushing 'a' onto the stack

    while (i < input.length() && input[i] == 'a') {

        pdaStack.push('a');

        i++;

    }

    // Transition for popping 'a' from the stack for each 'b'

    while (i < input.length() && input[i] == 'b') {

        if (pdaStack.empty()) {

            return false; // More 'b's than 'a's

        }

        pdaStack.pop();

        i++;

    }

    // Check if the input is fully processed and the stack is empty

    return i == input.length() && pdaStack.empty();

}

// Main function

int main() {

    string input;

    cout << "Enter a string over {a, b}: ";

    cin >> input;

    if (simulatePDA(input)) {

        cout << "String is accepted by the PDA." << endl;

    } else {

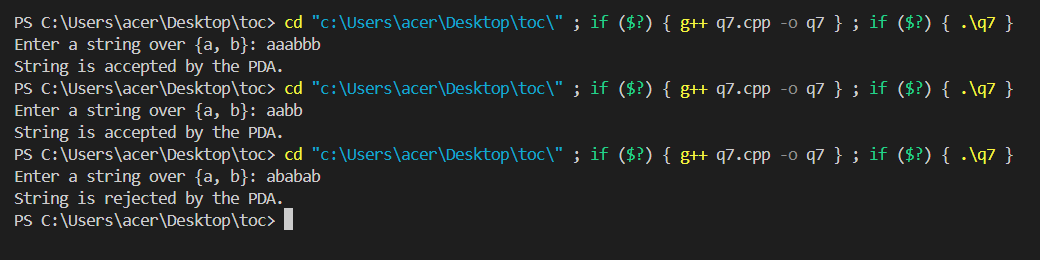
        cout << "String is rejected by the PDA." << endl;

    }

    return 0;

}

**Output:**



**Que.8**

Design a PDA and write a program for simulating the machine which accepts the language {w X w^r | w is any string over S={a, b} and w^r is reverse of that string and X is a special symbol }.

**Code Snippet:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Function to simulate PDA

bool simulatePDA(const string &input) {

    stack<char> pdaStack;

    int i = 0;

    // Step 1: Push the first part of the string (w) onto the stack

    while (i < input.length() && input[i] != 'X') {

        if (input[i] == 'a' || input[i] == 'b') {

            pdaStack.push(input[i]);

        } else {

            return false; // Invalid character before X

        }

        i++;

    }

    // Step 2: Ensure the special symbol 'X' is present

    if (i >= input.length() || input[i] != 'X') {

        return false; // No 'X' found

    }

    i++; // Skip 'X'

    // Step 3: Pop from the stack and match the second part of the string (wr)

    while (i < input.length()) {

        if (pdaStack.empty()) {

            return false; // More characters in wr than in w

        }

        char top = pdaStack.top();

        pdaStack.pop();

        if (input[i] != top) {

            return false; // Mismatch between stack and wr

        }

        i++;

    }

    // Step 4: Ensure the stack is empty after processing

    return pdaStack.empty();

}

// Main function

int main() {

    string input;

    cout << "Enter a string over {a, b, X}: ";

    cin >> input;

    if (simulatePDA(input)) {

        cout << "String is accepted by the PDA." << endl;

    } else {

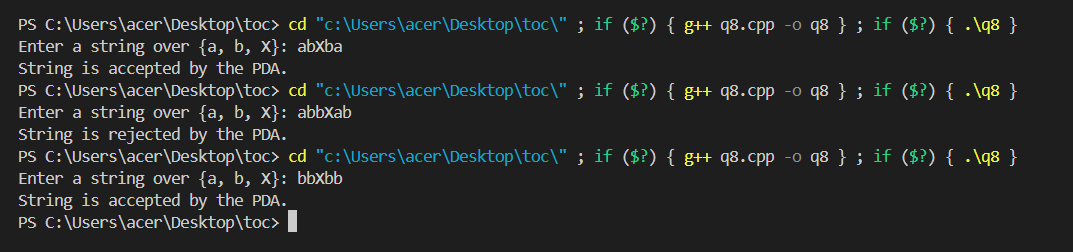
        cout << "String is rejected by the PDA." << endl;

    }

    return 0;

}

**Output:**

****

**Que.9**

Design and simulate a Turing Machine that accepts the language a^n b^n c^n where n >0.

**Code Snippet:**

#include <iostream>

#include <string>

using namespace std;

// Function to simulate the Turing Machine

bool turingMachine(string tape) {

    int n = tape.length();

    int i = 0;

    // Step 1: Ensure the input only contains a, b, c

    for (char ch : tape) {

        if (ch != 'a' && ch != 'b' && ch != 'c') {

            return false; // Invalid character

        }

    }

    while (true) {

        // Mark an 'a' with 'X'

        while (i < n && tape[i] != 'a') i++;

        if (i == n) break; // No more 'a' left to process

        tape[i] = 'X'; // Mark the 'a'

        i++;

        // Match with a 'b' and mark it as 'Y'

        while (i < n && tape[i] != 'b') i++;

        if (i == n || tape[i] != 'b') return false; // 'b' not found

        tape[i] = 'Y'; // Mark the 'b'

        i++;

        // Match with a 'c' and mark it as 'Z'

        while (i < n && tape[i] != 'c') i++;

        if (i == n || tape[i] != 'c') return false; // 'c' not found

        tape[i] = 'Z'; // Mark the 'c'

        i = 0; // Reset to start for the next iteration

    }

    // Check if all characters are marked

    for (char ch : tape) {

        if (ch != 'X' && ch != 'Y' && ch != 'Z') {

            return false; // Unprocessed symbols remain

        }

    }

    return true; // Successfully processed the tape

}

int main() {

    string input;

    cout << "Enter the string (a^n b^n c^n): ";

    cin >> input;

    if (turingMachine(input)) {

        cout << "Accepted: The string is in the language a^n b^n c^n\n";

    } else {

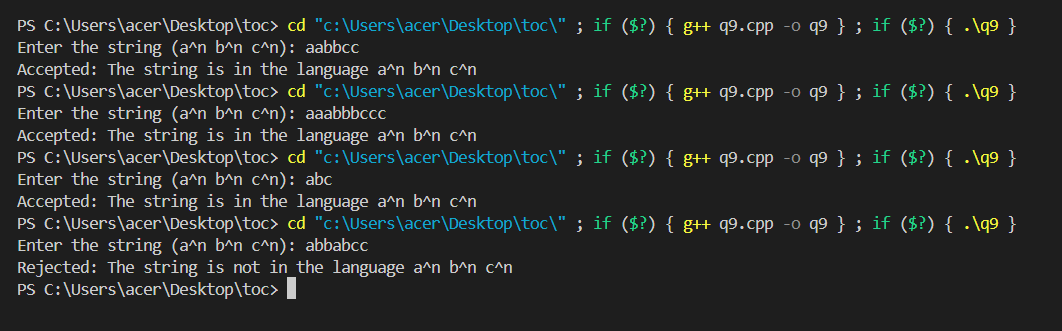
        cout << "Rejected: The string is not in the language a^n b^n c^n\n";

    }

    return 0;

}

**Output:**



**Que.10**

Design and simulate a Turing Machine which will increment the given binary number by 1.

**Code Snippet:**

#include <iostream>

#include <string>

using namespace std;

// Function to check if the input is a valid binary number

bool isValidBinary(const string& tape) {

    for (char ch : tape) {

        if (ch != '0' && ch != '1') {

            return false; // Invalid character found

        }

    }

    return true;

}

// Function to simulate the Turing Machine for binary increment

string turingMachineIncrement(string tape) {

    int n = tape.length();

    int i = n - 1; // Start from the rightmost bit

    // Traverse the tape to increment the binary number

    while (i >= 0) {

        if (tape[i] == '0') {

            tape[i] = '1'; // Flip 0 to 1

            return tape;   // No carry, return the result

        } else if (tape[i] == '1') {

            tape[i] = '0'; // Flip 1 to 0 and propagate the carry

            i--;           // Move left

        }

    }

    // If carry propagates beyond the leftmost bit

    return "1" + tape;

}

int main() {

    string input;

    cout << "Enter a binary number: ";

    cin >> input;

    // Validate the input binary number

    if (!isValidBinary(input)) {

        cout << "Error: Invalid binary input!" << endl;

    } else {

        string result = turingMachineIncrement(input);

        cout << "Incremented binary number: " << result << endl;

    }

    return 0;

}

**Output:**

