Questions :

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.

**C++ Code:**

#include <iostream>

#include <string>

using namespace std;

// Function prototypes

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);

int main() {

    string w;

    cout << "Enter a binary string: ";

    cin >> w;

    State0(w, 0);

    return 0;

}

void State0(const string &w, int i) {

    cout << "State 0" << endl;

    if (i == w.length()) {

        cout << "String is rejected (ending in non-final state)" << endl;

        return;

    }

    if (w[i] == '1') {

        State1(w, i + 1);

    } else if (w[i] == '0') {

        State0(w, i + 1);

    }

}

void State1(const string &w, int i) {

    cout << "State 1" << endl;

    if (i == w.length()) {

        cout << "String is rejected (ending in non-final state)" << endl;

        return;

    }

    if (w[i] == '1') {

        State2(w, i + 1);

    } else if (w[i] == '0') {

        State0(w, i + 1);

    }

}

void State2(const string &w, int i) {

    cout << "State 2" << endl;

    if (i == w.length()) {

        cout << "String is rejected (ending in non-final state)" << endl;

        return;

    }

    if (w[i] == '1') {

        State3(w, i + 1);

    } else if (w[i] == '0') {

        State0(w, i + 1);

    }

}

void State3(const string &w, int i) {

    cout << "State 3" << endl;

    if (i == w.length()) {

        cout << "String is accepted (contains '111')" << endl;

        return;

    }

    // Once in State3, stay in State3 regardless of input

    State3(w, i + 1);

}

**OUTPUT:**

**A computer screen with many colorful text

Description automatically generated**

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.

**C++ Code:**

#include <iostream>

#include <string>

using namespace std;

// Function prototypes

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);

int main() {

    string w;

    cout << "Enter a binary string: ";

    cin >> w;

    State0(w, 0);

    return 0;

}

void State0(const string &w, int i) {

    cout << "State 0" << endl;

    if (i == w.length()) {

        cout << "String is rejected (ending in non-final state)" << endl;

        return;

    }

    if (w[i] == '1') {

        State1(w, i + 1);

    } else if (w[i] == '0') {

        State0(w, i + 1);

    }

}

void State1(const string &w, int i) {

    cout << "State 1" << endl;

    if (i == w.length()) {

        cout << "String is rejected (ending in non-final state)" << endl;

        return;

    }

    if (w[i] == '1') {

        State2(w, i + 1);

    } else if (w[i] == '0') {

        State1(w, i + 1);

    }

}

void State2(const string &w, int i) {

    cout << "State 2" << endl;

    if (i == w.length()) {

        cout << "String is accepted (exactly two 1's)" << endl;

        return;

    }

    if (w[i] == '1') {

        State3(w, i + 1);

    } else if (w[i] == '0') {

        State2(w, i + 1);

    }

}

void State3(const string &w, int i) {

    cout << "State 3" << endl;

    if (i == w.length()) {

        cout << "String is accepted (exactly three 1's)" << endl;

        return;

    }

    if (w[i] == '1') {

        State4(w, i + 1);

    } else if (w[i] == '0') {

        State3(w, i + 1);

    }

}

void State4(const string &w, int i) {

    cout << "State 4" << endl;

    if (i == w.length()) {

        cout << "String is rejected (more than three 1's)" << endl;

        return;

    }

    // Remain in State4 regardless of input

    State4(w, i + 1);

}

**OUPUT:**

**A screen shot of a computer program

Description automatically generated**

1. 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.

**C++ Code:**

#include <iostream>

#include <string>

using namespace std;

void State0(const string &w, int i, char first1, char first2);

void State1(const string &w, int i, char first1, char first2);

void State2(const string &w, int i, char first1, char first2);

void State3(const string &w, int i, char first1, char first2);

int main() {

    string w;

    cout << "Enter a string: ";

    cin >> w;

    if (w.length() < 4) {

        cout << "String is rejected (length < 4)" << endl;

        return 0;

    }

    // Start the process by checking the first two characters

    State0(w, 0, w[0], w[1]);

    return 0;

}

void State0(const string &w, int i, char first1, char first2) {

    cout << "State 0" << endl;

    if (i == w.length()) {

        cout << "String is rejected (not long enough to check first and last two characters)" << endl;

        return;

    }

    // Transition to State1 to check the next character

    if (w[i] == 'a') {

        State1(w, i + 1, first1, first2);

    } else if (w[i] == 'b') {

        State1(w, i + 1, first1, first2);

    }

}

void State1(const string &w, int i, char first1, char first2) {

    cout << "State 1" << endl;

    if (i == w.length()) {

        cout << "String is rejected (not long enough to check first and last two characters)" << endl;

        return;

    }

    // Transition to State2 to process the next character

    if (w[i] == 'a') {

        State2(w, i + 1, first1, first2);

    } else if (w[i] == 'b') {

        State2(w, i + 1, first1, first2);

    }

}

void State2(const string &w, int i, char first1, char first2) {

    cout << "State 2" << endl;

    if (i == w.length()) {

        cout << "String is rejected (length < 4)" << endl;

        return;

    }

    // Proceed to State3 only when reaching last two characters

    if (i == w.length() - 2) {

        State3(w, i, first1, first2);  // Compare first and last two characters

    } else {

        // Continue the process for intermediate characters

        if (w[i] == 'a') {

            State2(w, i + 1, first1, first2);

        } else if (w[i] == 'b') {

            State2(w, i + 1, first1, first2);

        }

    }

}

void State3(const string &w, int i, char first1, char first2) {

    cout << "State 3" << endl;

    if (w[i] == first1 && w[i + 1] == first2) {

        cout << "String is accepted (first two characters match last two)" << endl;

    } else {

        cout << "String is rejected (mismatch in first and last two characters)" << endl;

    }

}

**OUTPUT:**

**A screen shot of a computer program

Description automatically generated**

1. 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.

**C++ Code**:

#include <iostream>

#include <string>

using namespace std;

void State0(const string &w, int i);

void State1(const string &w, int i);

void State2(const string &w, int i);

int main() {

    string w;

    cout << "Enter a string: ";

    cin >> w;

    // Check the minimum length constraint

    if (w.length() < 2) {

        cout << "String is rejected (length < 2)" << endl;

        return 0;

    }

    State0(w, 0);  // Start the finite automaton from State 0

    return 0;

}

void State0(const string &w, int i) {

    cout << "State 0" << endl;

    if (w[i] == 'a') {

        State1(w, i + 1);  // If 'a', move to State 1

    } else {

        cout << "String is rejected (starts with a character other than 'a')" << endl;

    }

}

void State1(const string &w, int i) {

    cout << "State 1" << endl;

    if (i == w.length() - 1) {

        // If last character is 'b', move to State 2

        if (w[i] == 'b') {

            State2(w, i + 1);

        } else {

            cout << "String is rejected (doesn't end with 'b')" << endl;

        }

    } else {

        // Process middle characters, stay in State 1

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

            State1(w, i + 1);

        } else {

            cout << "String is rejected (invalid character)" << endl;

        }

    }

}

void State2(const string &w, int i) {

    cout << "State 2" << endl;

    if (i == w.length()) {

        cout << "String is accepted (first character 'a' and last character 'b')" << endl;

    } else {

        cout << "String is rejected (invalid character in the middle)" << endl;

    }

}

**OUTPUT:** **A computer screen with many lines of text

Description automatically generated**

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

**C++ Code:**

#include <iostream>

#include <string>

using namespace std;

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);

int main() {

    string w;

    cout << "Enter a string: ";

    cin >> w;

    if (w.length() % 2 != 0) {

        cout << "String is rejected (length is odd)" << endl;

        return 0;

    }

    State0(w, 0); // Start from state 0

    return 0;

}

void State0(const string &w, int i) {

    cout << "State 0" << endl;

    if (i == w.length()) {

        cout << "String is accepted (both counts even)" << endl;

        return;

    }

    if (w[i] == 'a') {

        State1(w, i + 1); // Transition to state 1

    } else if (w[i] == 'b') {

        State2(w, i + 1); // Transition to state 2

    } else {

        cout << "Invalid character" << endl;

    }

}

void State1(const string &w, int i) {

    cout << "State 1" << endl;

    if (i == w.length()) {

        cout << "String is rejected (odd a's, even b's)" << endl;

        return;

    }

    if (w[i] == 'a') {

        State0(w, i + 1); // Transition to state 0

    } else if (w[i] == 'b') {

        State3(w, i + 1); // Transition to state 3

    } else {

        cout << "Invalid character" << endl;

    }

}

void State2(const string &w, int i) {

    cout << "State 2" << endl;

    if (i == w.length()) {

        cout << "String is rejected (even a's, odd b's)" << endl;

        return;

    }

    if (w[i] == 'a') {

        State3(w, i + 1); // Transition to state 3

    } else if (w[i] == 'b') {

        State0(w, i + 1); // Transition to state 0

    } else {

        cout << "Invalid character" << endl;

    }

}

void State3(const string &w, int i) {

    cout << "State 3" << endl;

    if (i == w.length()) {

        cout << "String is rejected (odd a's, odd b's)" << endl;

        return;

    }

    if (w[i] == 'a') {

        State2(w, i + 1); // Transition to state 2

    } else if (w[i] == 'b') {

        State1(w, i + 1); // Transition to state 1

    } else {

        cout << "Invalid character" << endl;

    }

}

**OUTPUT:** **A screen shot of a computer program

Description automatically generated**

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)

**C++ Code:**

#include <iostream>

#include <string>

// Function to check if a string belongs to L1

bool isInL1(const std::string &str) {

    if (str.length() >= 4) {

        // Check if the first two characters are the same as the last two

        return str.substr(0, 2) == str.substr(str.length() - 2);

    }

    return false;

}

// Function to check if a string belongs to L2

bool isInL2(const std::string &str) {

    if (str.length() >= 2 && str[0] == 'a' && str.back() == 'b') {

        // Check that all intermediate characters are either 'a' or 'b'

        for (size\_t i = 1; i < str.length() - 1; ++i) {

            if (str[i] != 'a' && str[i] != 'b') {

                return false;

            }

        }

        return true;

    }

    return false;

}

// Function to check if a string belongs to the union of L1 and L2

bool isInUnion(const std::string &str) {

    return isInL1(str) || isInL2(str);

}

// Function to check if a string belongs to the intersection of L1 and L2

bool isInIntersection(const std::string &str) {

    return isInL1(str) && isInL2(str);

}

// Function to check if a string belongs to the concatenation of L1 and L2

bool isInConcatenation(const std::string &str) {

    // Concatenation requires dividing the string into two substrings

    // Check each possible split point

    for (size\_t i = 1; i < str.length(); ++i) {

        std::string part1 = str.substr(0, i);

        std::string part2 = str.substr(i);

        if (isInL1(part1) && isInL2(part2)) {

            return true;

        }

    }

    return false;

}

int main() {

    std::string input;

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

    std::cin >> input;

    std::cout << "String is in L1: " << (isInL1(input) ? "Yes" : "No") << "\n";

    std::cout << "String is in L2: " << (isInL2(input) ? "Yes" : "No") << "\n";

    std::cout << "String is in Union (L1 U L2): " << (isInUnion(input) ? "Yes" : "No") << "\n";

    std::cout << "String is in Intersection (L1 ∩ L2): " << (isInIntersection(input) ? "Yes" : "No") << "\n";

    std::cout << "String is in Concatenation (L1 L2): " << (isInConcatenation(input) ? "Yes" : "No") << "\n";

    return 0;

}

**OUTPUT:** **A screen shot of a computer program

Description automatically generated**

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

**C++ Code:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Function to simulate the PDA

bool simulatePDA(const string &w) {

    stack<char> pdaStack;

    pdaStack.push('Z'); // Push initial stack symbol

    int i = 0;

    while (i < w.length()) {

        char current = w[i];

        if (current == 'a') {

            pdaStack.push('A'); // Push 'A' for each 'a'

        } else if (current == 'b') {

            if (!pdaStack.empty() && pdaStack.top() == 'A') {

                pdaStack.pop(); // Pop 'A' for each 'b'

            } else {

                return false; // Invalid sequence

            }

        } else {

            return false; // Invalid character in input

        }

        i++;

    }

    // Check if the stack is in a valid accept state

    return pdaStack.size() == 1 && pdaStack.top() == 'Z';

}

int main() {

    string w;

    cout << "Enter a string (a^n b^n, n > 0): ";

    cin >> w;

    if (simulatePDA(w)) {

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

    } else {

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

    }

    return 0;

}

**OUTPUT:**

A screen shot of a computer code

Description automatically generated

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

**C++ Code:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

// Function to simulate the PDA

bool simulatePDA(const string &w) {

    stack<char> pdaStack;

    int i = 0;

    // Push phase: Push characters of w onto the stack

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

        pdaStack.push(w[i]);

        i++;

    }

    // Check for the presence of 'X'

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

        return false; // Reject if 'X' is missing

    }

    i++; // Move past 'X'

    // Pop phase: Verify w^r by popping the stack

    while (i < w.length()) {

        if (pdaStack.empty() || pdaStack.top() != w[i]) {

            return false; // Reject if the stack doesn't match

        }

        pdaStack.pop();

        i++;

    }

    // Accept if stack is empty and input is fully processed

    return pdaStack.empty();

}

int main() {

    string w;

    cout << "Enter a string (wXw^r): ";

    cin >> w;

    if (simulatePDA(w)) {

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

    } else {

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

    }

    return 0;

}

**OUTPUT:**

**A screen shot of a computer code

Description automatically generated**

9. Design and simulate a Turing Machine that accepts the language anbncn where n >0.

**C++ Code:**

#include <iostream>

#include <string>

#include <vector>

bool simulate\_anbncn(const std::string& input) {

    // Validate string format

    if (input.empty()) return false;

    // Count a, b, c occurrences

    int countA = 0, countB = 0, countC = 0;

    // First pass: count a's

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

        countA++;

    // Second pass: count b's

    int j = countA;

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

        countB++;

        j++;

    }

    // Third pass: count c's

    while (j < input.length() && input[j] == 'c') {

        countC++;

        j++;

    }

    // Validation conditions

    return (countA > 0 &&

            countA == countB &&

            countB == countC &&

            j == input.length());

}

int main() {

    // Test cases

    std::vector<std::string> testCases = {

        "abc",         // Valid

        "aabbcc",      // Valid

        "aaabbbccc",   // Valid

        "aabcc",       // Invalid

        "abbc",        // Invalid

        "",            // Invalid

        "aaa"          // Invalid

    };

    for (const auto& test : testCases) {

        std::cout << "Input: " << test

                  << " - "

                  << (simulate\_anbncn(test) ? "Accepted" : "Rejected")

                  << std::endl;

    }

    return 0;

}

**OUTPUT:**

**A black screen with text

Description automatically generated**

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

**C++ Code:**

#include <iostream>

#include <string>

std::string incrementBinary(std::string input) {

    // Pad with leading zero if needed

    input = '0' + input;

    // Traverse from right to left

    for (int i = input.length() - 1; i >= 0; i--) {

        if (input[i] == '0') {

            input[i] = '1';

            break;

        } else if (input[i] == '1') {

            input[i] = '0';

        }

    }

    // Remove leading zeros manually

    int leadingZeros = 0;

    while (leadingZeros < input.length() - 1 && input[leadingZeros] == '0') {

        leadingZeros++;

    }

    input = input.substr(leadingZeros);

    return input.empty() ? "0" : input;

}

int main() {

    const char\* testCases[] = {"0", "1", "10", "11", "1010", "1111"};

    int numCases = sizeof(testCases) / sizeof(testCases[0]);

    for (int i = 0; i < numCases; i++) {

        std::string result = incrementBinary(testCases[i]);

        std::cout << testCases[i] << " + 1 = " << result << std::endl;

    }

    return 0;

}

**OUTPUT:**

**A black screen with white text

Description automatically generated**