






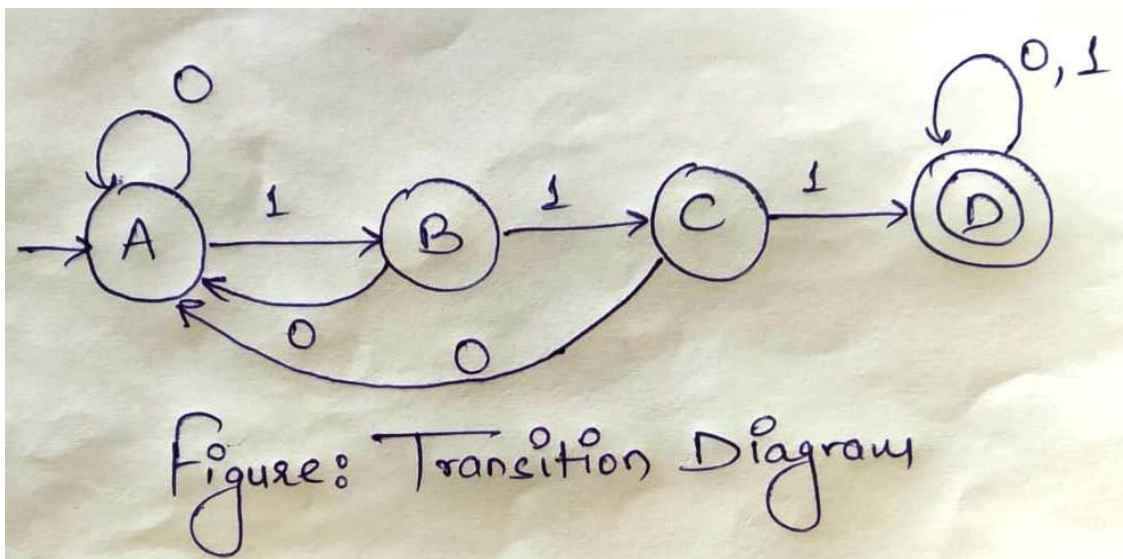
THEORY OF COMPUTATION

PRCTICAL FILE



-  NAME – TUSHAR
-  ROLL NUMBER – 22/28087
-  COURSE – BSC(HONS)COMPUTER
SCIENCE
-  SEMESTER – 5TH
-  SUBMITTED TO – DR. SHALINI
AGGARWAL MADAM

Q1. 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.



```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  bool acceptsThreeConsecutiveOnes(const string& input) {
7      int state = 0; // Start at state q0
8
9      for (char ch : input) {
10         switch (state) {
11             case 0: // q0
12                 if (ch == '0') {
13                     state = 0; // Stay in q0
14                 } else if (ch == '1') {
15                     state = 1; // Move to q1
16                 }
17                 break;
18             case 1: // q1
19                 if (ch == '0') {
20                     state = 0; // Go back to q0
21                 } else if (ch == '1') {
22                     state = 2; // Move to q2
23                 }
24                 break;
25             case 2: // q2
26                 if (ch == '0') {
27                     state = 0; // Go back to q0
28                 } else if (ch == '1') {
29                     state = 3; // Move to q3 (accepting)
30                 }
31                 break;
32             case 3: // q3 (accepting)
33                 // Remain in accepting state on '0' or '1'
34                 state = 3; // Stay in q3
35                 break;
36         }
37     }
38
39     // Check if we are in an accepting state
40     return (state == 3 || state == 4);
41 }

```

```

42
43 int main() {
44     string input;
45
46     cout << "Enter a binary string: ";
47     getline(cin, input);
48
49     if (acceptsThreeConsecutiveOnes(input)) {
50         cout << "Accepted: The string contains '111'." << endl;
51     } else {
52         cout << "Rejected: The string does not contain '111'." << endl;
53     }
54
55     return 0; // Indicating successful completion of the program
56 }

```

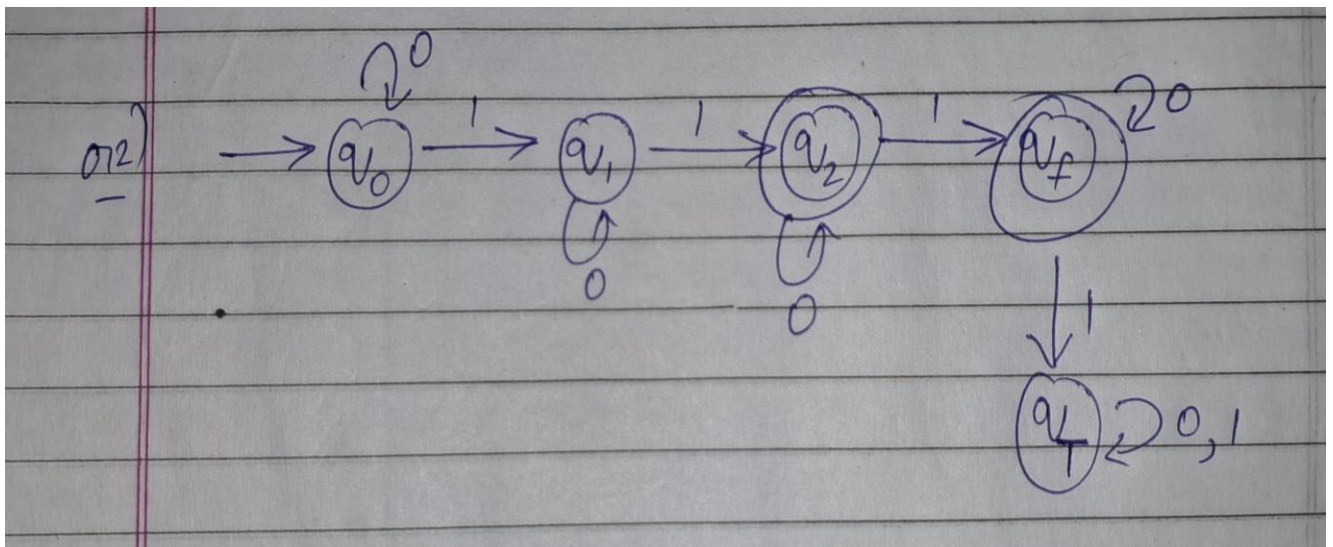
TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a binary string: 011010010101
Rejected: The string does not contain '111'.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a binary string: 0111111111000010
Accepted: The string contains '111'.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a binary string: 110110110110
Rejected: The string does not contain '111'.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a binary string: 0000110010001010010100101110
Accepted: The string contains '111'.

```

Q2. 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.



```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  bool acceptsTwoOrThreeOnes(const string& input) {
5      int state = 0; // Start at state q0
6      int oneCount = 0; // Count of '1's
7
8      for (char ch : input) {
9          switch (state) {
10             case 0: // q0
11                 if (ch == '0') {
12                     state = 0; // Stay in q0
13                 } else if (ch == '1') {
14                     state = 1; // Move to q1
15                     oneCount++;
16                 }
17                 break;
18             case 1: // q1
19                 if (ch == '0') {
20                     state = 1; // Stay in q1
21                 } else if (ch == '1') {
22                     state = 2; // Move to q2
23                     oneCount++;
24                 }
25                 break;
26             case 2: // q2
27                 if (ch == '0') {
28                     state = 2; // Stay in q2
29                 } else if (ch == '1') {
30                     state = 3; // Move to q3
31                     oneCount++;
32                 }
33                 break;
34             case 3: // q3 (accepting state)
35                 if (ch == '0') {
36                     state = 3; // Stay in q3
37                 } else if (ch == '1') {
38                     state = 4; // Move to q4 (dead state)
39                 }
40                 break;
41             case 4: // q4 (dead state)
42                 // Stay in dead state on any input
43                 break;
44             }
45         }
46         // The string is accepted if we end in q2 or q3
47         return (state == 2 || state == 3);
48     }
49
50 int main() {
51     string input;
52
53     cout << "Enter a binary string: ";
54     getline(cin, input);
55
56     if (acceptsTwoOrThreeOnes(input)) {
57         cout << "Accepted: The string contains exactly two or three '1's." << endl;
58     } else {
59         cout << "Rejected: The string does not contain exactly two or three '1's." << endl;
60     }
61     return 0; // Indicating successful completion of the program
62 }

```

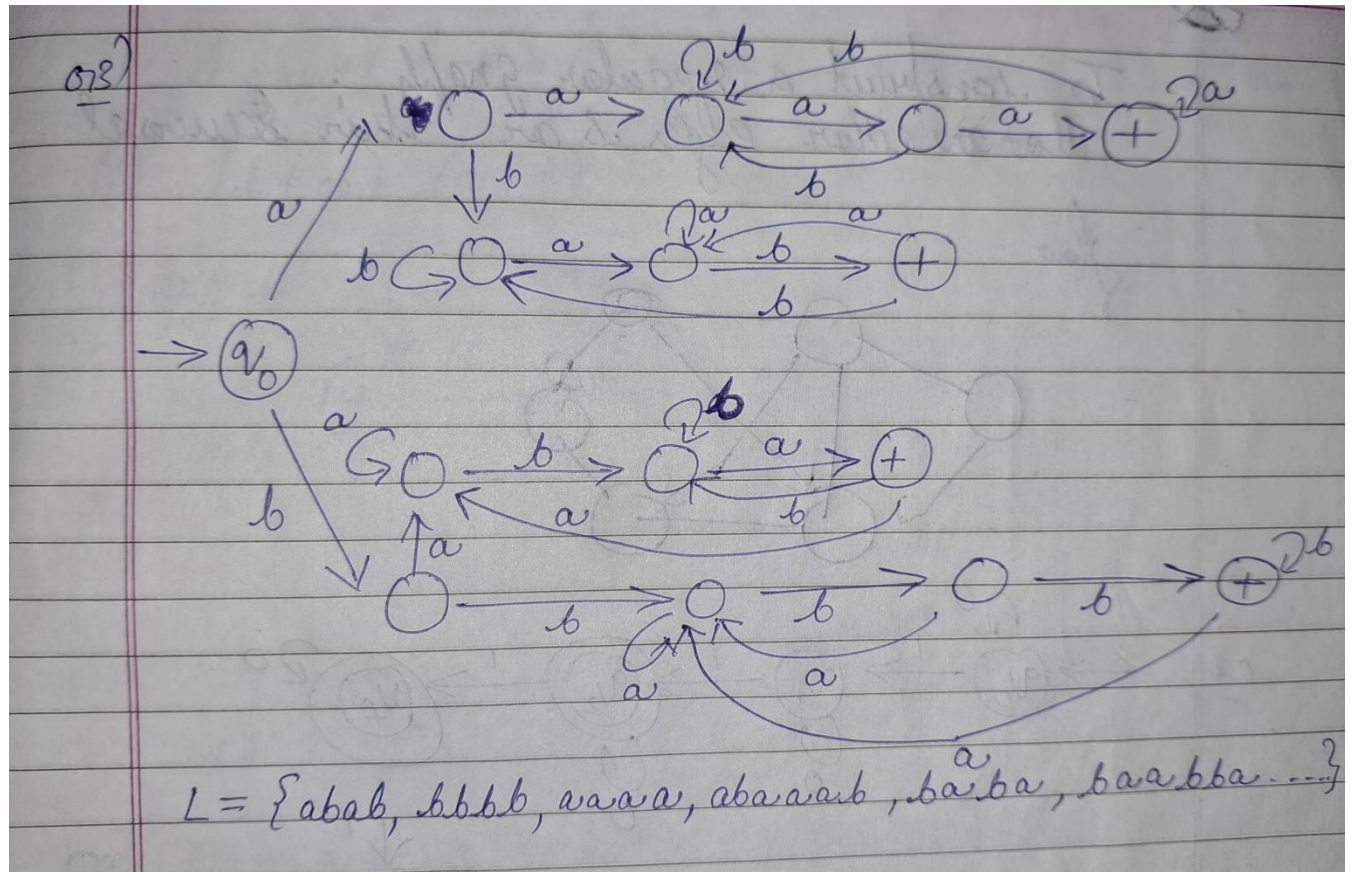
TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop
Enter a binary string: 01101010
Rejected: The string does not contain exactly two or three '1's.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop
Enter a binary string: 01100010
Accepted: The string contains exactly two or three '1's.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop
Enter a binary string: 100001000
Accepted: The string contains exactly two or three '1's.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop
Enter a binary string: 011111110001001001
Rejected: The string does not contain exactly two or three '1's.

```


Q3. Design a Finite Automata (FA) that accepts language L_1 , 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.



```

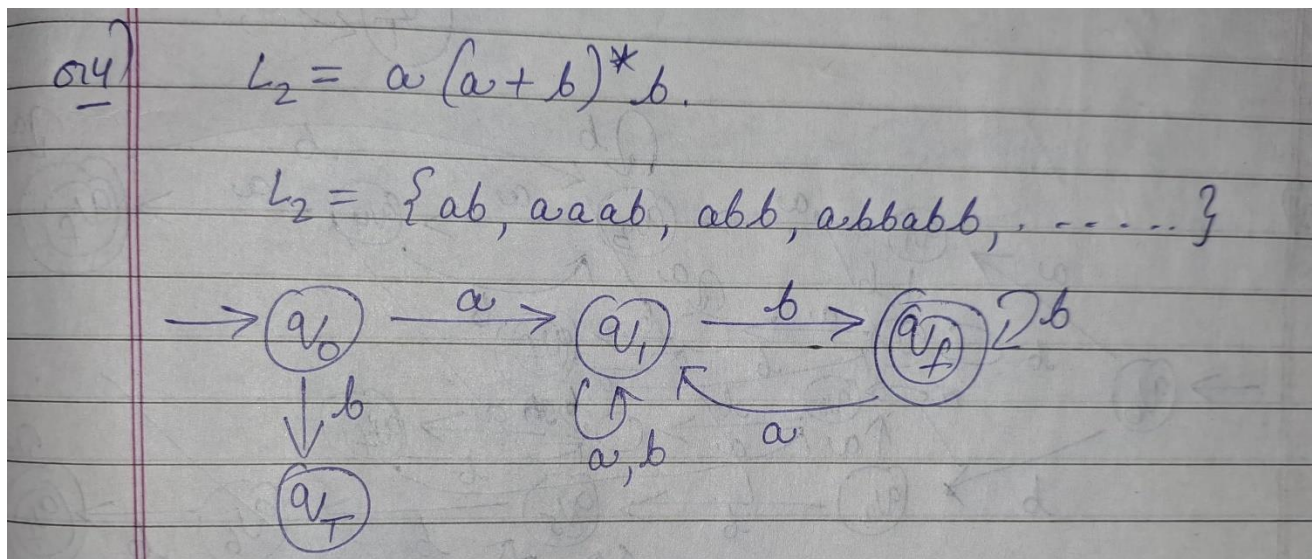
1 #include <iostream>
2 #include <string>
3
4 bool acceptsL1(const std::string &input) {
5     int length = input.length();
6     // Check if length is less than 4, immediately reject
7     if (length < 4) return false;
8
9     // Check if the first two characters match the last two
10    return (input[0] == input[length - 2] && input[1] == input[length - 1]);
11 }
12
13 int main() {
14     std::string input;
15     std::cout << "Enter a string over {a, b}: ";
16     std::cin >> input;
17
18     if (acceptsL1(input)) {
19         std::cout << "The string is accepted by the FA.\n";
20     } else {
21         std::cout << "The string is rejected by the FA.\n";
22     }
23
24     return 0;
25 }

```

TERMINAL OUTPUT:

```
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a string over {a, b}: abababbababbbbababab
The string is accepted by the FA.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a string over {a, b}: baba
The string is accepted by the FA.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a string over {a, b}: bbbb
The string is accepted by the FA.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a string over {a, b}: aaaaaa
The string is accepted by the FA.
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
Enter a string over {a, b}: babbbababbbbabbbabb
The string is rejected by the FA.
```

Q4. Design a Finite Automata (FA) that accepts language L_2 , over $S = \{a, b\}$ where $L_2 = a(a+b)^*b$. Write a program to simulate this FA.



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 // Function to simulate the FA
5 bool simulateFA(const string &input) {
6     // Define states
7     enum State { q0, q1, q2 };
8     State currentState = q0;
9
10    // Process each character in the input string
11    for (char ch : input) {
12        switch (currentState) {
13            case q0:
14                if (ch == 'a') {
15                    currentState = q1; // Move to q1 on initial 'a'
16                } else {
17                    return false; // Reject if it doesn't start with 'a'
18                }
19                break;
20
21            case q1:
22                if (ch == 'a') {
23                    currentState = q1; // Stay in q1 on 'a'
24                } else if (ch == 'b') {
25                    currentState = q2; // Move to q2 on 'b'
26                }
27                break;
28
29            case q2:
30                if (ch == 'a') {
31                    currentState = q1; // Return to q1 on 'a'
32                } else if (ch == 'b') {
33                    currentState = q2; // Stay in q2 on 'b'
34                }
35                break;
36        }
37    }
38    return currentState == q2;
39 }
```

```

35         break;
36     }
37 }
38 // Accept if the final state is q2
39 return (currentState == q2);
40 }
41 int main() {
42     // Test cases
43     string testStrings[] = { "ab", "aab", "abb", "aaabb", "abab", "abba", "a", "b", "ababbab", "abaab", "aababab" };
44     for (const string &test : testStrings) {
45         cout << "Input: " << test << " => "
46             << (simulateFA(test) ? "Accepted" : "Rejected") << endl;
47     }
48     return 0;
49 }

```

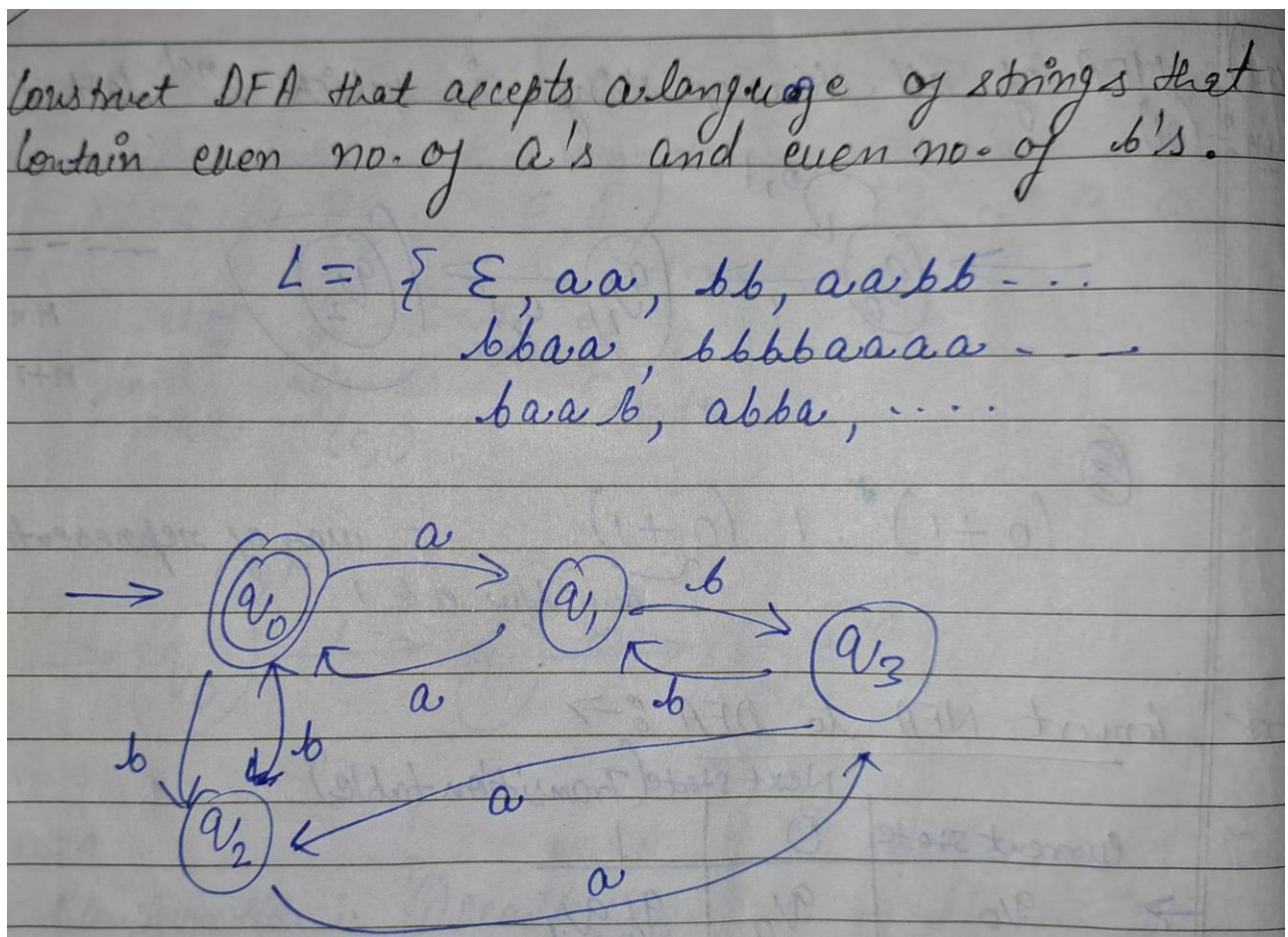
TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\T
Input: ab => Accepted
Input: aab => Accepted
Input: abb => Accepted
Input: aaabb => Accepted
Input: abab => Accepted
Input: abba => Rejected
Input: a => Rejected
Input: b => Rejected
Input: ababbab => Accepted
Input: abaab => Accepted
Input: aababab => Accepted

```

Q5. Design a Finite Automata (FA) that accepts language EVEN-EVEN over $S=\{a, b\}$. Write a program to simulate this FA.




```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // Function to simulate the FA for EVEN-EVEN
7  bool simulateEvenEvenFA(const string &input) {
8      // Define states
9      enum State { q_ee, q_eo, q_oe, q_oo };
10     State currentState = q_ee;
11
12     // Process each character in the input string
13     for (char ch : input) {
14         switch (currentState) {
15             case q_ee:
16                 if (ch == 'a') {
17                     currentState = q_oe;
18                 } else if (ch == 'b') {
19                     currentState = q_eo;
20                 }
21                 break;
22
23             case q_eo:
24                 if (ch == 'a') {
25                     currentState = q_oo;
26                 } else if (ch == 'b') {
27                     currentState = q_ee;
28                 }
29                 break;
30
31             case q_oe:
32                 if (ch == 'a') {
33                     currentState = q_ee;
34                 } else if (ch == 'b') {
35                     currentState = q_oo;
36                 }
37                 break;
38
39             case q_oo:
40                 if (ch == 'a') {
41                     currentState = q_eo;
42                 } else if (ch == 'b') {
43                     currentState = q_oe;
44                 }
45                 break;
46         }
47     }
48
49     // Accept if the final state is q_ee
50     return (currentState == q_ee);
51 }
52
53 int main() {
54     // Test cases
55     string testStrings[] = { "", "aa", "bb", "aabb", "abab", "abba", "a", "b", "ab", "ba", "bbb", "aaabbbb" };
56     for (const string &test : testStrings) {
57         cout << "Input: " << test << " => "
58             << (simulateEvenEvenFA(test) ? "Accepted" : "Rejected") << endl;
59     }
60     return 0;
61 }

```

TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theo
Input:  => Accepted
Input: aa => Accepted
Input: bb => Accepted
Input: aabb => Accepted
Input: abab => Accepted
Input: abba => Accepted
Input: a => Rejected
Input: b => Rejected
Input: ab => Rejected
Input: ba => Rejected
Input: bbb => Rejected
Input: aaabbbb => Rejected

```


Q6. Write a program to simulate an FA that accepts

A. Union of the languages L1 and L2.

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // FA for L1: Accepts strings that start with 'a' and end with 'b'
7  bool simulateFA_L1(const string &input) {
8      if (input.length() < 2) return false; // Minimum length for "a...b" is 2
9      return (input[0] == 'a' && input[input.length() - 1] == 'b');
10 }
11
12 // FA for L2: Accepts strings that contain "aa" as a substring
13 bool simulateFA_L2(const string &input) {
14     for (size_t i = 0; i < input.length() - 1; ++i) {
15         if (input[i] == 'a' && input[i + 1] == 'a') {
16             return true;
17         }
18     }
19     return false;
20 }
21
22 // Union of L1 and L2: Accept if either L1 or L2 accepts the string
23 bool simulateFA_Union(const string &input) {
24     return simulateFA_L1(input) || simulateFA_L2(input);
25 }
26
27 int main() {
28     // Test cases
29     string testStrings[] = { "ab", "aab", "aa", "bbaa", "ba", "aabb", "aaab", "bb", "aaaaab" };
30     for (const string &test : testStrings) {
31         cout << "Input: " << test << " => "
32         << (simulateFA_Union(test) ? "Accepted" : "Rejected") << endl;
33     }
34     return 0;
35 }
```

TERMINAL OUTPUT:

```
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\
Input: ab => Accepted
Input: aab => Accepted
Input: aa => Accepted
Input: bbaa => Accepted
Input: ba => Rejected
Input: aabb => Accepted
Input: aaab => Accepted
Input: bb => Rejected
Input: aaaaab => Accepted
```

B. Intersection of the languages L1 and L2

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // FA for L1: Accepts strings that start with 'a' and end with 'b'
7  bool simulateFA_L1(const string &input) {
8      if (input.length() < 2) return false; // Minimum length for "a...b" is 2
9      return (input[0] == 'a' && input[input.length() - 1] == 'b');
10 }
11
12 // FA for L2: Accepts strings that contain "aa" as a substring
13 bool simulateFA_L2(const string &input) {
14     for (size_t i = 0; i < input.length() - 1; ++i) {
15         if (input[i] == 'a' && input[i + 1] == 'a') {
16             return true;
17         }
18     }
19     return false;
20 }
21
22 // Intersection of L1 and L2: Accept if both L1 and L2 accept the string
23 bool simulateFA_Intersection(const string &input) {
24     return simulateFA_L1(input) && simulateFA_L2(input);
25 }
26
27 int main() {
28     // Test cases
29     string testStrings[] = { "ab", "aab", "aa", "aabb", "aaab", "aba", "ba", "bbaa", "abababbbb", "abaababbbbbbbab" };
30     for (const string &test : testStrings) {
31         cout << "Input: " << test << " => "
32             << (simulateFA_Intersection(test) ? "Accepted" : "Rejected") << endl;
33     }
34     return 0;
35 }
```

TERMINAL OUTPUT:

```
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tu
Input: ab => Rejected
Input: aab => Accepted
Input: aa => Rejected
Input: aabb => Accepted
Input: aaab => Accepted
Input: aba => Rejected
Input: ba => Rejected
Input: bbaa => Rejected
Input: abababbbb => Rejected
Input: abaababbbbbbbab => Accepted
```

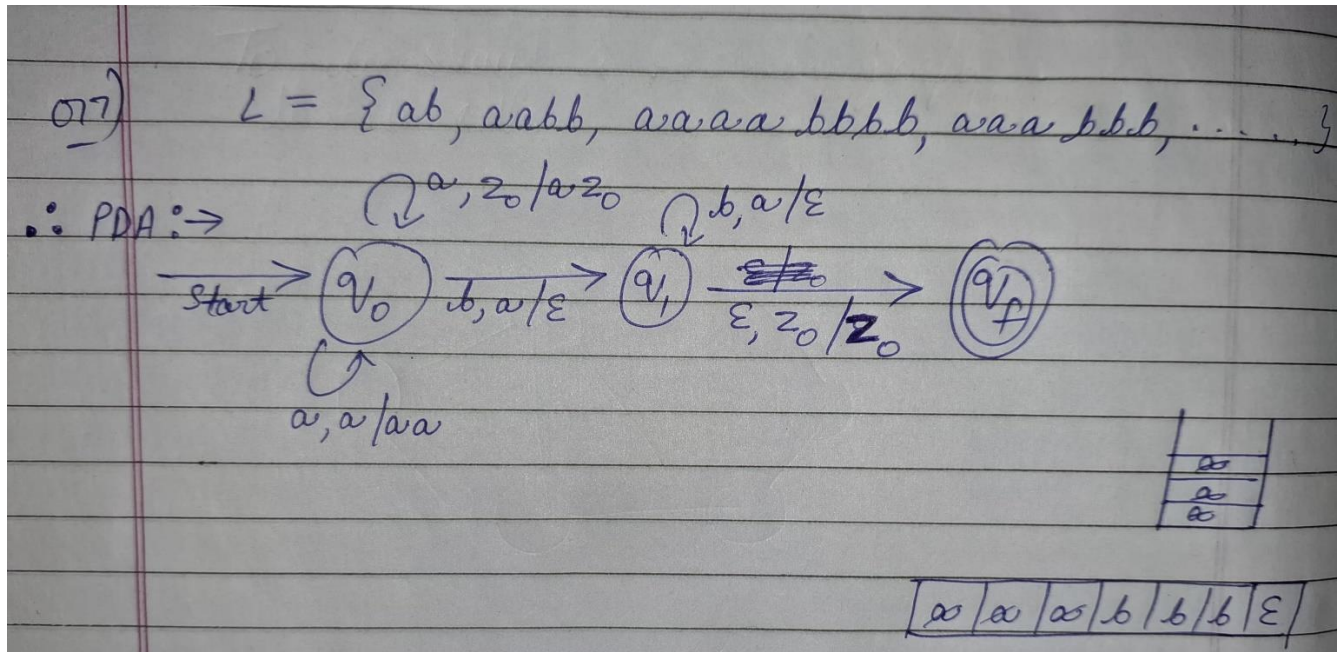
C. Language L1 L2 (concatenation)

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // FA for L1: Accepts strings that start with 'a' and end with 'b'
7  bool simulateFA_L1(const string &input) {
8      if (input.length() < 2) return false; // Minimum length for "a...b" is 2
9      return (input[0] == 'a' && input[input.length() - 1] == 'b');
10 }
11
12 // FA for L2: Accepts strings that contain "aa" as a substring
13 bool simulateFA_L2(const string &input) {
14     for (size_t i = 0; i < input.length() - 1; ++i) {
15         if (input[i] == 'a' && input[i + 1] == 'a') {
16             return true;
17         }
18     }
19     return false;
20 }
21
22 // Concatenation of L1 and L2: Accept if input can be split such that
23 // the first part is accepted by L1 and the second part is accepted by L2
24 bool simulateFA_Concatenation(const string &input) {
25     for (size_t i = 1; i < input.length(); ++i) {
26         string part1 = input.substr(0, i); // First part for L1
27         string part2 = input.substr(i);    // Second part for L2
28         if (simulateFA_L1(part1) && simulateFA_L2(part2)) {
29             return true;
30         }
31     }
32     return false;
33 }
34
35 int main() {
36     // Test cases
37     string testStrings[] = { "ab", "aab", "aabaa", "aabba", "aabaaa", "aaab", "abb", "aaabb", "abbabbbabaab", "aabbabab" };
38     for (const string &test : testStrings) {
39         cout << "Input: " << test << " => "
40             << (simulateFA_Concatenation(test) ? "Accepted" : "Rejected") << endl;
41     }
42     return 0;
43 }
```

TERMINAL OUTPUT:

```
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\
Input: ab => Rejected
Input: aab => Rejected
Input: aabaa => Accepted
Input: aabba => Rejected
Input: aabaaa => Accepted
Input: aaab => Rejected
Input: abb => Rejected
Input: aaabb => Rejected
Input: abbabbbabaab => Accepted
Input: aabbabab => Rejected
```

Q7. Design a PDA and write a program for simulating the machine which accepts the language $\{a^n b^n \text{ where } n > 0, S = \{a, b\}\}$.



```

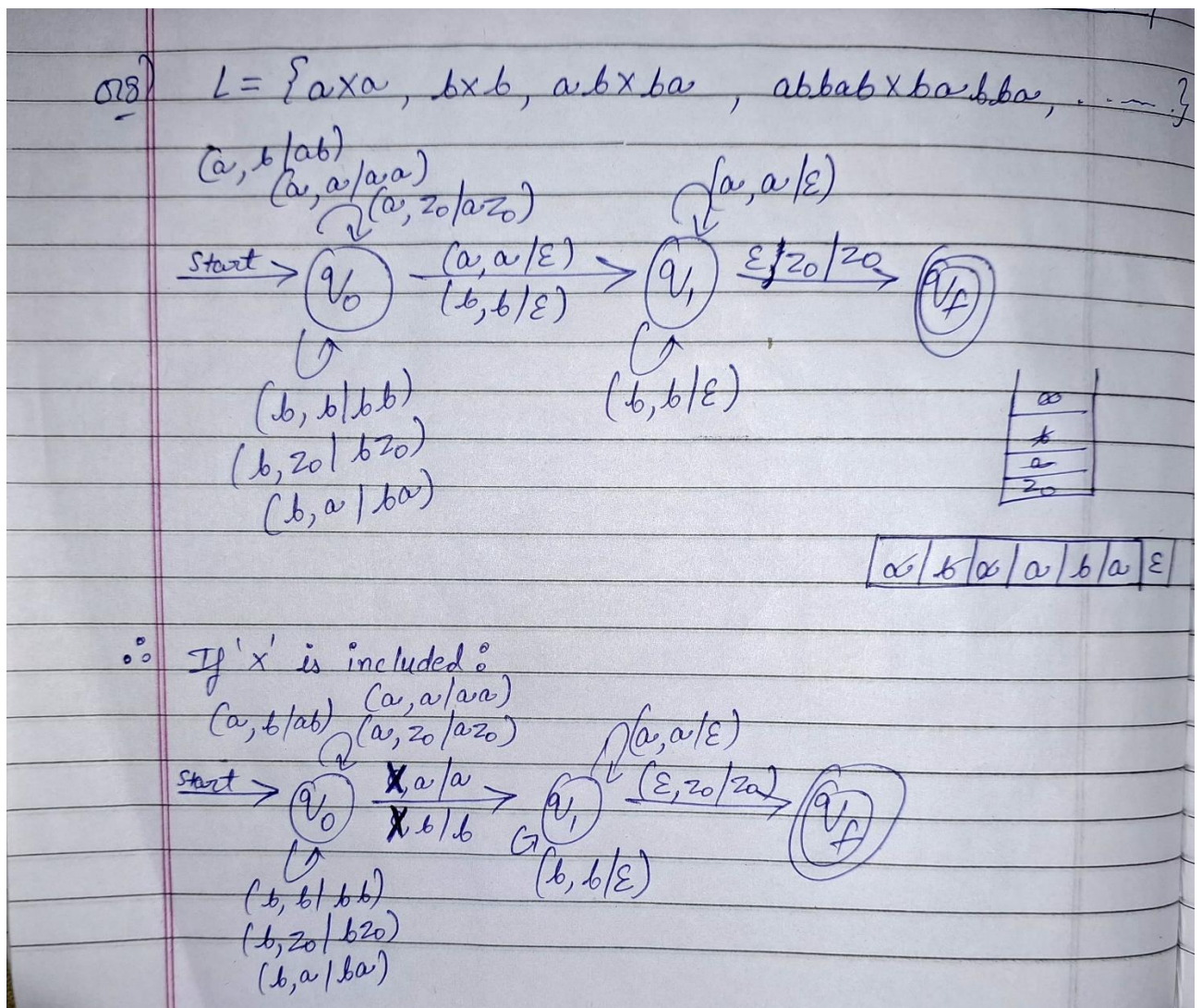
1  #include <iostream>
2  #include <stack>
3  #include <string>
4
5  using namespace std;
6
7  // Function to simulate the PDA
8  bool simulatePDA(const string &input) {
9      stack<char> pdaStack;
10
11      // Step 1: Push 'A' onto the stack for each 'a' in the input
12      size_t i = 0;
13      while (i < input.length() && input[i] == 'a') {
14          pdaStack.push('A'); // Push a marker for each 'a'
15          i++;
16      }
17
18      // Step 2: Pop 'A' from the stack for each 'b' in the input
19      while (i < input.length() && input[i] == 'b') {
20          if (pdaStack.empty()) {
21              return false; // More 'b's than 'a's, so reject
22          }
23          pdaStack.pop(); // Pop a marker for each 'b'
24          i++;
25      }
26
27      // Step 3: Check if the stack is empty and we have processed the entire input
28      return (pdaStack.empty() && i == input.length());
29  }
30
31  int main() {
32      // Test cases
33      string testStrings[] = { "ab", "aabb", "aaabbb", "aaaabbbb", "aaabb", "aabbb", "abbb", "a" };
34      for (const string &test : testStrings) {
35          cout << "Input: " << test << " => "
36              << (simulatePDA(test) ? "Accepted" : "Rejected") << endl;
37      }
38      return 0;
39  }

```


TERMINAL OUTPUT:

```
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha"
Input: ab => Accepted
Input: aabb => Accepted
Input: aaabbb => Accepted
Input: aaaabbbb => Accepted
Input: aaabb => Rejected
Input: aabbb => Rejected
Input: abbb => Rejected
Input: a => Rejected
```

Q8. Design a PDA and write a program for simulating the machine which accepts the language $\{wXwr \mid w \text{ is any string over } S = \{a, b\} \text{ and } wr \text{ is reverse of that string and } X \text{ is a special symbol}\}$.



```

1  #include <iostream>
2  #include <stack>
3  #include <string>
4
5  using namespace std;
6
7  // Function to simulate the PDA
8  bool simulatePDA(const string &input) {
9      stack<char> pdaStack;
10     size_t i = 0;
11
12     // Step 1: Push characters of `w` onto the stack until we reach 'X'
13     while (i < input.length() && input[i] != 'X') {
14         pdaStack.push(input[i]);
15         i++;
16     }
17
18     // Step 2: Check for the presence of 'X' in the middle
19     if (i == input.length() || input[i] != 'X') {
20         return false; // Reject if there's no 'X' separating `w` and `w^r`
21     }
22
23     // Skip the 'X' character
24     i++;
25
26     // Step 3: Pop from the stack and match with `w^r`
27     while (i < input.length()) {
28         if (pdaStack.empty() || pdaStack.top() != input[i]) {
29             return false; // Mismatch or stack empty before finishing `w^r`
30         }
31         pdaStack.pop();
32         i++;
33     }
34
35     // Step 4: Accept if the stack is empty after processing all input
36     return pdaStack.empty();
37 }
38
39 int main() {
40     // Test cases
41     string testStrings[] = { "aXa", "abbaXabba", "abXba", "abbXaab", "bXb", "aabXbaa", "aaXaa", "aaabbaXabbaaa" };
42     for (const string &test : testStrings) {
43         cout << "Input: " << test << " => "
44             << (simulatePDA(test) ? "Accepted" : "Rejected") << endl;
45     }
46     return 0;
47 }

```

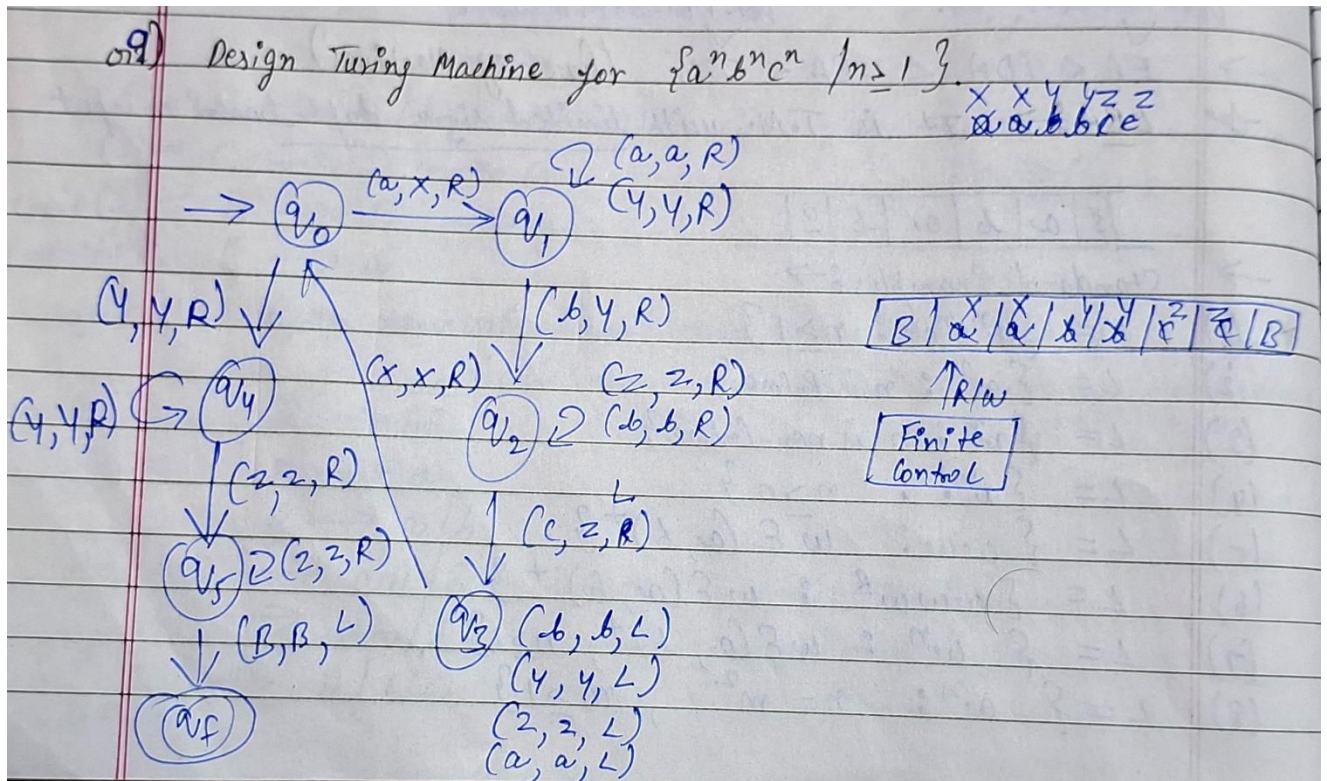
TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tu
Input: aXa => Accepted
Input: abbaXabba => Accepted
Input: abXba => Accepted
Input: abbXaab => Rejected
Input: bXb => Accepted
Input: aabXbaa => Accepted
Input: aaXaa => Accepted
Input: aaabbaXabbaaa => Accepted

```

Q9. Design and simulate a Turing Machine that accepts the language $a^n b^n c^n$ where $n > 0$.



```

1  #include <iostream>
2  #include <string>
3  bool simulateTM(std::string tape) {
4      int head = 0; // Start at the beginning of the tape
5
6      while (true) {
7          // Find the first unmarked 'a' and mark it as 'X'
8          head = tape.find('a');
9          if (head != std::string::npos) {
10             tape[head] = 'X';
11         } else {
12             // If there are no more 'a's, we should also have no 'b's or 'c's left
13             if (tape.find('b') == std::string::npos && tape.find('c') == std::string::npos) {
14                 return true; // Accepted
15             } else {
16                 return false; // Rejected due to mismatched count
17             }
18         }
19
20         // Find the first unmarked 'b' after marking 'a' and mark it as 'Y'
21         head = tape.find('b', head);
22         if (head != std::string::npos) {
23             tape[head] = 'Y';
24         } else {
25             return false; // Rejected if no matching 'b' is found for 'a'
26         }
27
28         // Find the first unmarked 'c' after marking 'b' and mark it as 'Z'
29         head = tape.find('c', head);
30         if (head != std::string::npos) {
31             tape[head] = 'Z';
32         } else {
33             return false; // Rejected if no matching 'c' is found for 'a' and 'b'
34         }
35     }
36 }
37

```



```

38 int main() {
39     std::string input;
40     std::cout << "Enter a string: ";
41     std::cin >> input;
42
43     if (simulateTM(input)) {
44         std::cout << "Accepted" << std::endl;
45     } else {
46         std::cout << "Rejected" << std::endl;
47     }
48
49     return 0;
50 }

```

TERMINAL OUTPUT:

```

PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha
Enter a string: abbbbbbcccc
Rejected
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha
Enter a string: abc
Accepted
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha
Enter a string: aaabbbccc
Accepted
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha
Enter a string: aabbbbaabbbac
Rejected

```

Q10. Design and simulate a Turing Machine which will increment the given binary number by 1.

Q10) Design a T.M. which will increment the given binary no. by 1.

Soln:→

Input:→ 0000
output:→ 0001

Input:→ 1111
output:→ 10000

Input:→ 0101
output:→ 0110

Input:→ 1001
output:→ 1010

```

graph LR
    start(( )) --> q0((q0))
    q0 -- "(0,0,R)" --> q0
    q0 -- "(1,1,R)" --> q1((q1))
    q1 -- "(1,0,L)" --> q1
    q1 -- "(B,1,N)" --> qf(((qf)))
    q1 -- "(0,1,N)" --> qf
    style start fill:none,stroke:none
    style qf fill:none,stroke:none

```

∴ R → shows move pointer to Right direction.

∴ L → shows move point to left direction.

∴ N → shows move head of machine to stay in the same place & halt.

Finite Control

↑ R/w

B	0	1	0	1	B
---	---	---	---	---	---


```

1  #include <iostream>
2  #include <string>
3  #include <algorithm>
4  class TuringMachine {
5  public:
6      TuringMachine(const std::string& input) : tape(input), head(0) {}
7
8      void increment() {
9          // Move to the end of the binary number
10         head = tape.size() - 1;
11
12         // State: Starting from the rightmost bit
13         while (head >= 0) {
14             if (tape[head] == '0') {
15                 // Change '0' to '1' and we're done
16                 tape[head] = '1';
17                 return;
18             } else {
19                 // Change '1' to '0' and carry over
20                 tape[head] = '0';
21                 head--;
22             }
23         }
24         // If we carried out from the leftmost bit, we need to add '1' at the start
25         tape = '1' + tape;
26     }
27
28     std::string getResult() const {
29         return tape;
30     }
31
32 private:
33     std::string tape; // The tape with the binary number
34     int head;        // The position of the head
35 };
36
37 int main() {
38     std::string binaryNumber;
39     std::cout << "Enter a binary number: ";
40     std::cin >> binaryNumber;
41
42     // Ensure the input is a valid binary number
43     if (binaryNumber.find_first_not_of("01") != std::string::npos) {
44         std::cerr << "Invalid binary number!" << std::endl;
45         return 1;
46     }
47
48     TuringMachine tm(binaryNumber);
49     tm.increment();
50     std::cout << "Incremented binary number: " << tm.getResult() << std::endl;
51     return 0;
52 }

```

TERMINAL OUTPUT:

```

Enter a binary number: 1101
Incremented binary number: 1110
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
}
Enter a binary number: 0
Incremented binary number: 1
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
}
Enter a binary number: 111
Incremented binary number: 1000
PS C:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation> cd "c:\Users\tusha\OneDrive\Desktop\CODING\c++\Theory_of_computation"
}
Enter a binary number: 0101
Incremented binary number: 0110

```