

PROGRAM

Name : Ankit Kumar

Course : B Sc (H) Computer Science

Roll no : 16005

Subject: THEORY OF COMPUTATION

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.

```
#include <iostream>
```

```
using namespace std;
```

```
void State0(string w, int i);
```

```
void State1(string w, int i);
```

```
void State2(string w, int i);
```

```
void State3(string w, int i);
```

```
int main() {
```

```
    string w; // User-entered string
```

```
    cout << "Enter a string: ";
```

```
    cin >> w;
```

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

```
    return 0;
```

```
}
```

```
void State0(string w, int i) {
```

```
    cout << "State 0" << endl;
```

```
    if (i == w.length()) {
```

```
        cout << "String is rejected" << endl; // Rejected, did not reach State3
```

```

        return;
    }
    if (w[i] == '1')
        State1(w, i + 1); // Transition to State1 on '1'
    else
        State0(w, i + 1); // Stay in State0 on '0'
}

void State1(string w, int i) {
    cout << "State 1" << endl;
    if (i == w.length()) {
        cout << "String is rejected" << endl; // Rejected, did not reach State3
        return;
    }
    if (w[i] == '1')
        State2(w, i + 1); // Transition to State2 on another '1'
    else
        State0(w, i + 1); // Reset to State0 on '0'
}

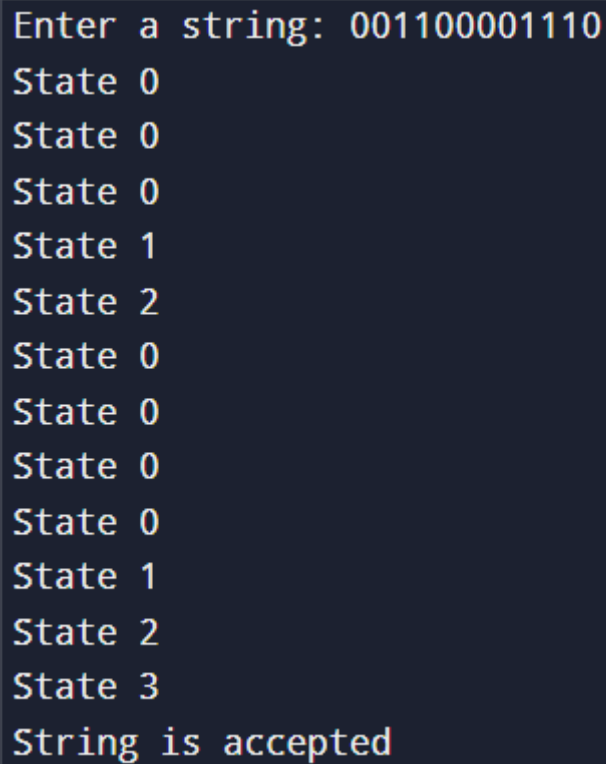
void State2(string w, int i) {
    cout << "State 2" << endl;
    if (i == w.length()) {
        cout << "String is rejected" << endl; // Rejected, did not reach State3
        return;
    }
    if (w[i] == '1')
        State3(w, i + 1); // Transition to State3 on a third '1'
    else
        State0(w, i + 1); // Reset to State0 on '0'
}

```

```

void State3(string w, int i) {
    cout << "State 3" << endl;
    cout << "String is accepted" << endl; // String contains three consecutive '1's
}

```



```

Enter a string: 001100001110
State 0
State 0
State 0
State 1
State 2
State 0
State 0
State 0
State 0
State 1
State 2
State 3
String is accepted

```

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.

```

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

```

```

void State0(string w, int i);
void State1(string w, int i);
void State2(string w, int i);
void State3(string w, int i);

```

```
void State4(string w, int i);
```

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

```
void State0(string w, int i) {  
    if (i == w.length()) {  
        cout << "String is rejected" << endl;  
        return;  
    }  
    if (w[i] == '1') {  
        State1(w, i + 1);  
    } else {  
        State0(w, i + 1); // Stay in State 0 for '0'  
    }  
}
```

```
void State1(string w, int i) {  
    if (i == w.length()) {  
        cout << "String is rejected" << endl;  
        return;  
    }  
    if (w[i] == '1') {  
        State2(w, i + 1);  
    } else {  
        State1(w, i + 1); // Stay in State 1 for '0'  
    }  
}
```

```
}  
}
```

```
void State2(string w, int i) {  
    if (i == w.length()) {  
        cout << "String is accepted" << endl; // Final state for exactly two 1's  
        return;  
    }  
    if (w[i] == '1') {  
        State3(w, i + 1);  
    } else {  
        State2(w, i + 1); // Stay in State 2 for '0'  
    }  
}
```

```
void State3(string w, int i) {  
    if (i == w.length()) {  
        cout << "String is accepted" << endl; // Final state for exactly three 1's  
        return;  
    }  
    if (w[i] == '1') {  
        State4(w, i + 1);  
    } else {  
        State3(w, i + 1); // Stay in State 3 for '0'  
    }  
}
```

```
void State4(string w, int i) {  
    if (i == w.length()) {  
        cout << "String is rejected" << endl; // Rejected state for more than three 1's  
        return;  
    }
```

```

    }
    if (w[i] == '1' || w[i] == '0') {
        State4(w, i + 1); // Stay in rejected state
    }
}

```

Output:

```

Enter a binary string: 000111000
State 0
State 0
State 0
State 0
State 1
State 2
State 3
State 3
State 3
State 3
String is accepted

```

Q3) 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.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
void State0(string w, int i, char first, char second);
```

```
void State1(string w, int i, char first, char second);
```

```
void State2(string w, int i, char first, char second);
```

```
void State3(string w, int i, char first, char second);
```

```
int main() {
```

```
    string w;
```

```

cout << "Enter a string over {a, b}: ";
cin >> w;
if (w.length() < 4) {
    cout << "String is rejected (length less than 4)." << endl;
} else {
    State0(w, 0, '\0', '\0'); // Start with State 0
}
return 0;
}

```

```

void State0(string w, int i, char first, char second) {
    if (i >= 2) {
        State1(w, i, first, second); // Transition to State 1 after capturing first two characters
    } else {
        if (i == 0) {
            first = w[i];
        } else if (i == 1) {
            second = w[i];
        }
        State0(w, i + 1, first, second); // Collect first two characters
    }
}

```

```

void State1(string w, int i, char first, char second) {
    if (i == w.length() - 2) {
        State2(w, i, first, second); // Move to State 2 to check the last two characters
    } else {
        State1(w, i + 1, first, second); // Keep traversing until the last two characters
    }
}

```

```

void State2(string w, int i, char first, char second) {
    if (w[i] == first && w[i + 1] == second) {
        State3(w, i, first, second); // Final state if last two characters match the first two
    } else {
        cout << "String is rejected" << endl;
        return;
    }
}

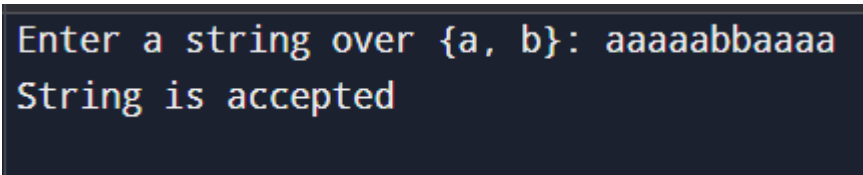
```

```

void State3(string w, int i, char first, char second) {
    cout << "String is accepted" << endl; // String satisfies the condition
    return;
}

```

Output:



```

Enter a string over {a, b}: aaaaabbbaaaa
String is accepted

```

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.

```

#include <iostream>

Using namespace std;

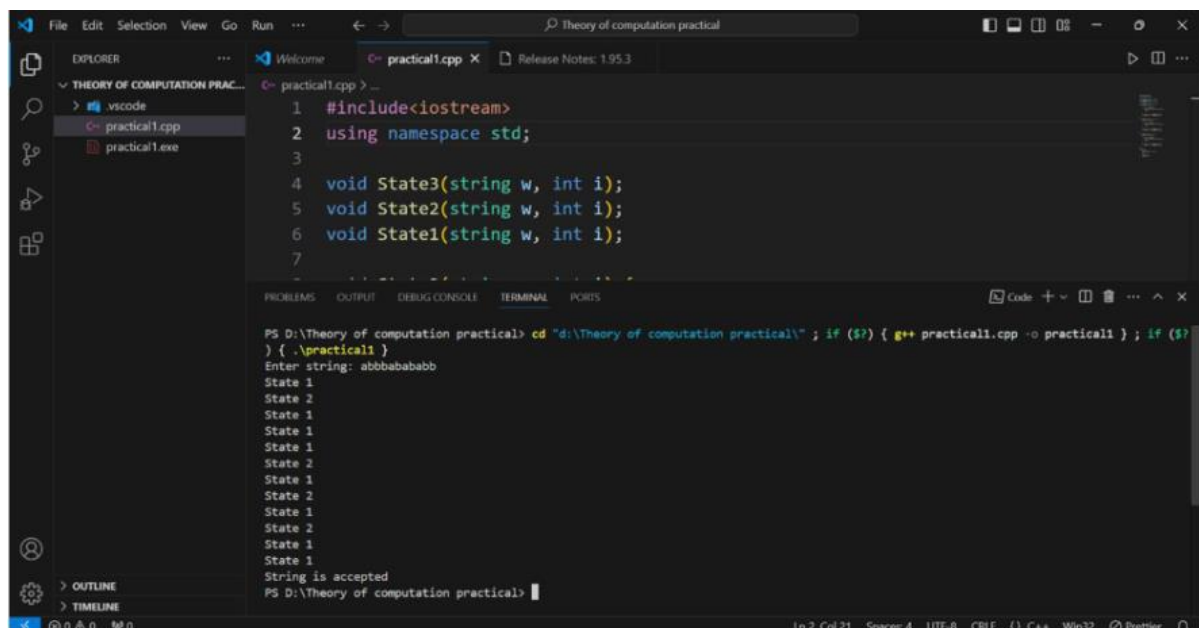
void State3(string w, int i);
void State2(string w, int i);
void State1(string w, int i);
void State3(string w, int i) {
    cout << "State 3\n";
    if (i == w.length()) {
        cout << "String is rejected\n";
        return;
    }
}

```



```
}  
if (w[i] == 'b') State3(w, i+1);  
if (w[i] == 'a') State3(w, i+1);  
}  
void State2(string w, int i) {  
    cout << "State 2\n";  
    if (i == w.length()) {  
        cout << "String is accepted\n";  
        return;  
    }  
    if (w[i] == 'b') State1(w, i+1);  
    if (w[i] == 'a') State3(w, i+1);  
}  
void State1(string w, int i) {  
    cout << "State 1\n";  
    if (i == w.length()) {  
        cout << "String is accepted\n";  
        return;  
    }  
    if (w[i] == 'b') State1(w, i+1);  
    if (w[i] == 'a') State2(w, i+1);  
}  
main() {  
    string w;  
    cout << "Enter string: ";  
    cin >> w;  
    State1(w, 0);  
    return 0;  
}
```

Output:



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'THEORY OF COMPUTATION PRAC...' containing 'practical1.cpp' and 'practical1.exe'. The Editor pane shows the code for 'practical1.cpp':

```
1 #include<iostream>
2 using namespace std;
3
4 void State3(string w, int i);
5 void State2(string w, int i);
6 void State1(string w, int i);
7
```

The TERMINAL pane at the bottom shows the command prompt output:

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical1.cpp -o practical1 } ; if ($?) { .\practical1 }
Enter string: abbbabababb
State 1
State 2
State 1
State 1
State 1
State 2
State 1
State 2
State 1
State 2
State 1
State 1
String is accepted
PS D:\Theory of computation practical>
```

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

```
#include <iostream>

Using namespace std;

void State1(string w, int i);

void State2(string w, int i);

void State3(string w, int i);

void State4(string w, int i);

void State4(string w, int i) {
    cout << "State 4\n";
    if (i == w.length()) {
        cout << "String is accepted\n";
        return;
    }
    if (w[i] == 'a') State3(w, i+1);
    if (w[i] == 'b') State1(w, i+1);
}
```

```

void State3(string w, int i) {
    cout << "State 3\n";
    if (i == w.length()) {
        cout << "String is rejected\n";
        return;
    }
    if (w[i] == 'a') State4(w, i+1);
    if (w[i] == 'b') State2(w, i+1);
}

void State2(string w, int i) {
    cout << "State 2\n";
    if (i == w.length()) {
        cout << "String is rejected\n";
        return;
    }
    if (w[i] == 'a') State1(w, i+1);
    if (w[i] == 'b') State3(w, i+1);
}

void State1(string w, int i) {
    cout << "State 1\n";
    if (i == w.length()) {
        cout << "String is rejected\n";
        return;
    }
    if (w[i] == 'a') State2(w, i+1);
    if (w[i] == 'b') State4(w, i+1);
}

main() {
    string w;
    cout << "Enter string: ";
    cin >> w;

```

```

State1(w, 0);

return 0;

}

```

Output:

```

// practical1.cpp
23 }
24
25 if (w[i] == 'a') State2(w, i+1);
26 if (w[i] == 'b') State4(w, i+1);
27 }
28
29 void State2(string w, int i) {
30     if (i == w.length()) {
31         cout << "String is rejected\n";
32         return;
33     }

```

```

PS D:\Theory of computation practical> cd "D:\Theory of computation practical\" ; if ($?) { g++ practical1.cpp -o practical1 } ; if ($?) { .practical1 }
Enter string: abab
String is rejected
PS D:\Theory of computation practical>

```

Q6) Write a program to simulate an FA that accepts

- Union of the languages L1 and L2
- Intersection of the languages L1 and L2
- Language L1 L2 (concatenation).

```

#include<iostream>

#include <string.h>

using namespace std;

int unionLanguage(char* w) {

int len = strlen(w);

if (w[0] == 'a') return 1;

if (w[len-1] == 'b') return 1;

return 0;

}

int intersectionLanguage(char* w) {

int len = strlen(w);

```

```

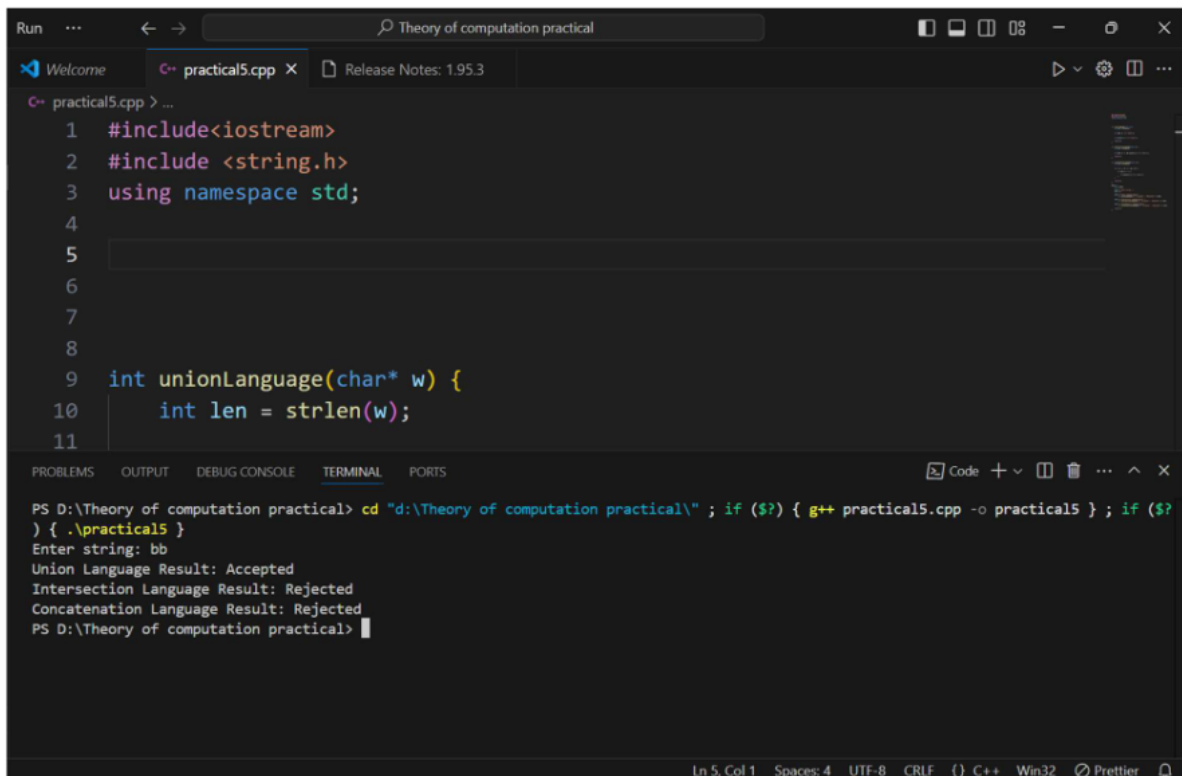
if (w[0] == 'a' && w[len-1] == 'b') return 1;
return 0;
}

int concatenationLanguage(char* w) {
int len = strlen(w);
for (int i = 0; i < len; i++) {
if (w[0] == 'a') {
if (w[len-1] == 'b') return 1;
}
}
return 0;
}

main() {
char w[100];
cout << "Enter string: ";
cin >> w;
cout << "Union Language Result: "
<< (unionLanguage(w) ? "Accepted" : "Rejected") << endl;
cout << "Intersection Language Result: "
<< (intersecitonLanguage(w) ? "Accepted" : "Rejected") << endl;
cout << "Concatenation Language Result: "
<< (concatenationLanguage(w) ? "Accepted" : "Rejected") << endl;
return 0;
}

```

Output:



The screenshot shows a Visual Studio Code window with a file named 'practical5.cpp' open. The code in the editor is as follows:

```
1 #include<iostream>
2 #include <string.h>
3 using namespace std;
4
5
6
7
8
9 int unionLanguage(char* w) {
10     int len = strlen(w);
11 }
```

The terminal at the bottom shows the execution of the program. The command executed is `cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }`. The output of the program is:

```
Enter string: bb
Union Language Result: Accepted
Intersection Language Result: Rejected
Concatenation Language Result: Rejected
PS D:\Theory of computation practical>
```

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

```
#include<iostream>

#include <stack>

#include <string.h>

using namespace std;

int simulatePDA(char* input) {

stack<char> s;

int i, len = strlen(input);

int a_count = 0, b_count = 0;

for (i = 0; input[i] != '\0'; i++) {

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

s.push('a');

a_count++;

} else if (input[i] == 'b') {

if (s.empty() || s.top() != 'a')
```

```

return 0;

s.pop();

b_count++;

}

}

return (a_count == b_count && s.empty());

}

main() {

char input[100];

cout << "Enter string: ";

cin >> input;

if (simulatePDA(input))

cout << "String accepted\n";

else

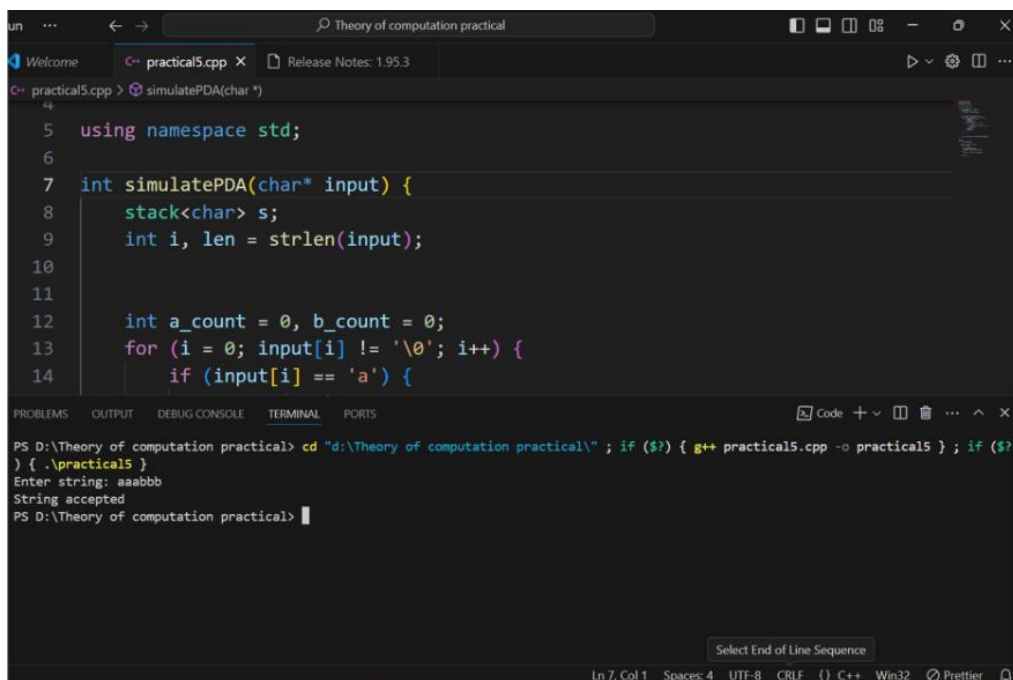
cout << "String rejected\n";

return 0;

}

```

Output:



The screenshot shows a Visual Studio Code window with a C++ file named `practical5.cpp`. The code defines a `simulatePDA` function that checks if a string is accepted by a Pushdown Automaton. The function uses a stack to count 'a's and 'b's. The `main` function prompts the user to enter a string and calls `simulatePDA`. The terminal output shows the program being compiled and executed, with the input string `aaabbb` and the output `String accepted`.

```

5  using namespace std;
6
7  int simulatePDA(char* input) {
8      stack<char> s;
9      int i, len = strlen(input);
10
11
12      int a_count = 0, b_count = 0;
13      for (i = 0; input[i] != '\0'; i++) {
14          if (input[i] == 'a') {

```

```

PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter string: aaabbb
String accepted
PS D:\Theory of computation practical>

```

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

```
#include<iostream>

#include <stack>

#include <string.h>

using namespace std;

int simulatePDA(char* input) {

    stack<char> s;

    int i, len = strlen(input);

    int a_count = 0, b_count = 0;

    for (i = 0; input[i] != '\0'; i++) {

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

            s.push('a');

            a_count++;

        } else if (input[i] == 'b') {

            if (s.empty() || s.top() != 'a')

                return 0;

            s.pop();

            b_count++;

        }

    }

    return (a_count == b_count && s.empty());

}

main() {

    char input[100];

    cout << "Enter string: ";

    cin >> input;

    if (simulatePDA(input))

        cout << "String accepted\n";

    else
```

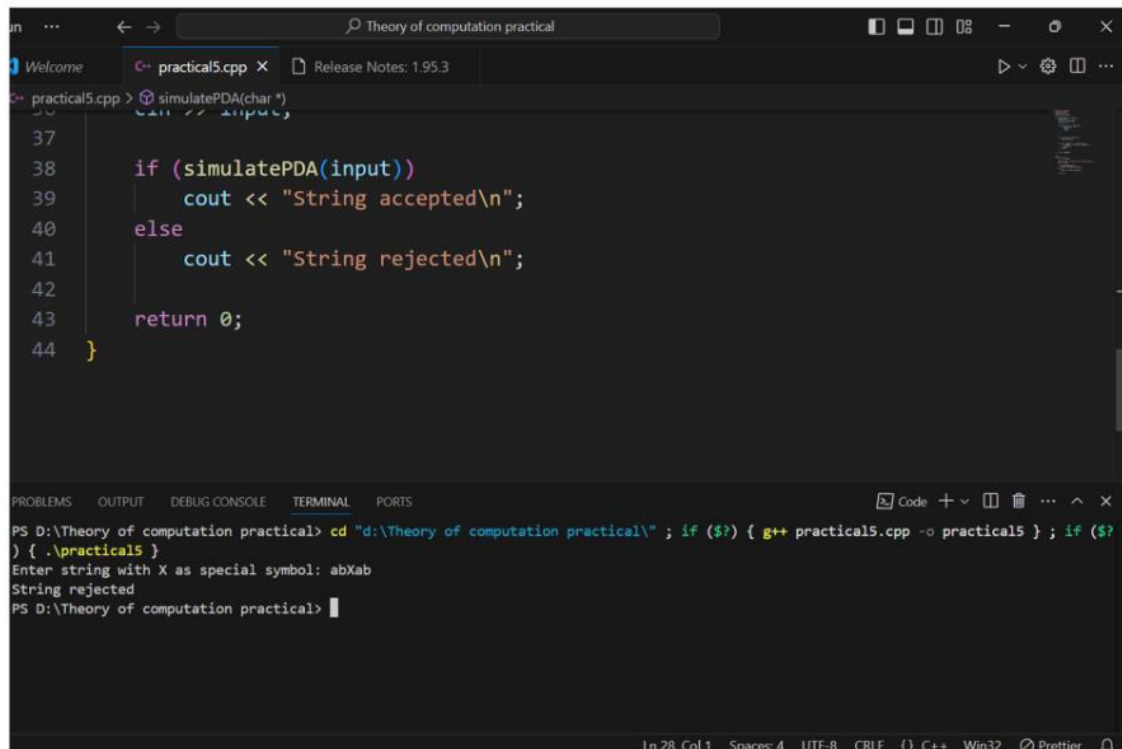


```
cout << "String rejected\n";

return 0;

}
```

Output:

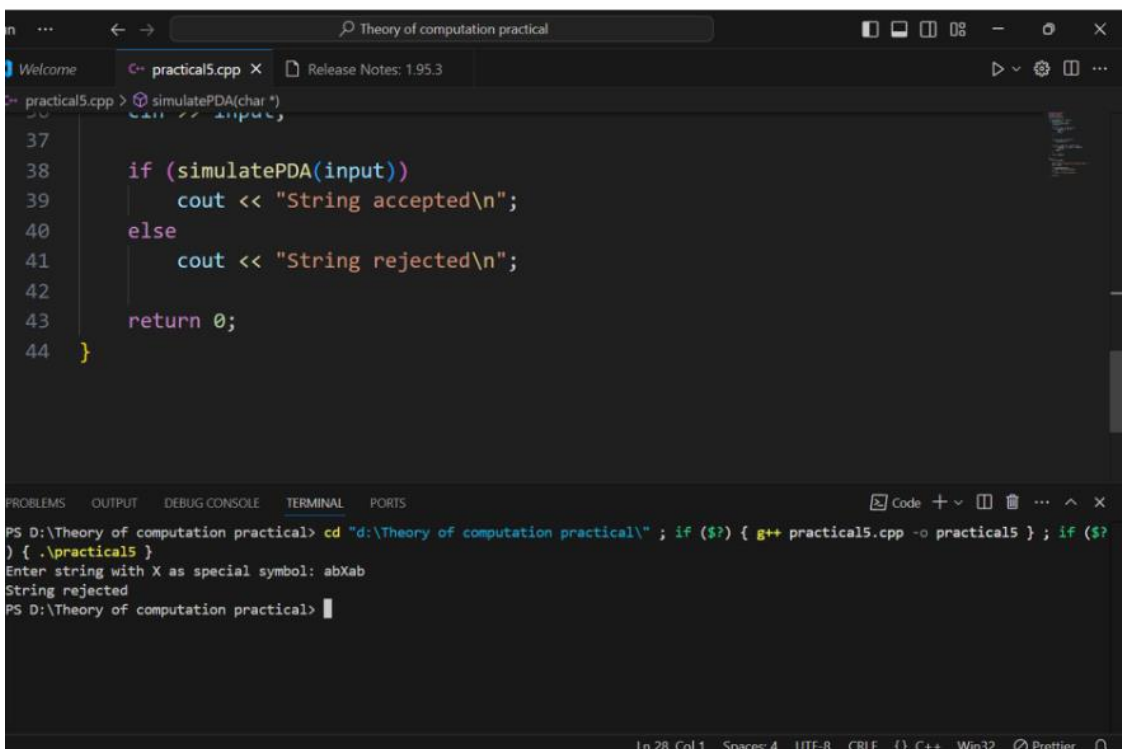


The screenshot shows a Visual Studio Code window with a file named 'practical5.cpp'. The code in the editor is as follows:

```
37
38     if (simulatePDA(input))
39         cout << "String accepted\n";
40     else
41         cout << "String rejected\n";
42
43     return 0;
44 }
```

Below the editor, the 'TERMINAL' tab is active, showing the command prompt output:

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter string with X as special symbol: abXab
String rejected
PS D:\Theory of computation practical>
```



This screenshot is identical to the one above, showing the same Visual Studio Code window with the 'practical5.cpp' file and the terminal output. The code in the editor is:

```
37
38     if (simulatePDA(input))
39         cout << "String accepted\n";
40     else
41         cout << "String rejected\n";
42
43     return 0;
44 }
```

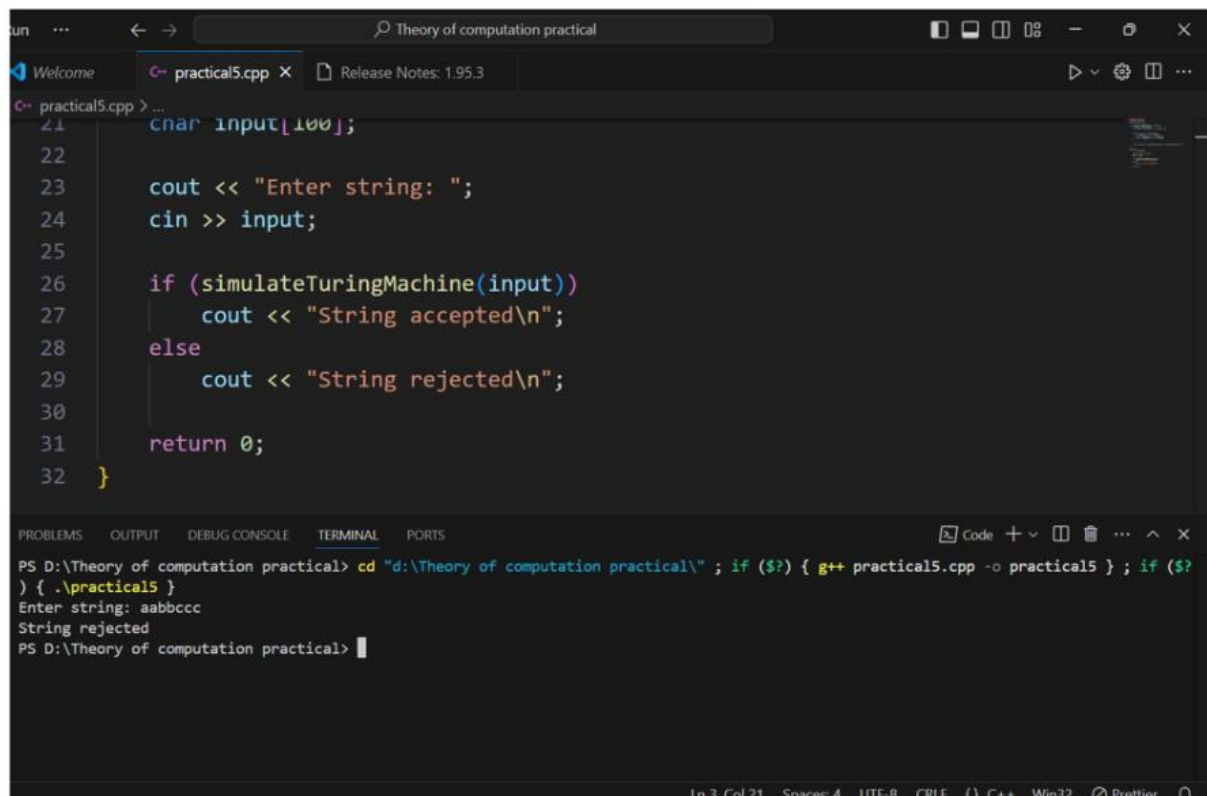
The terminal output is also identical:

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter string with X as special symbol: abXab
String rejected
PS D:\Theory of computation practical>
```

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

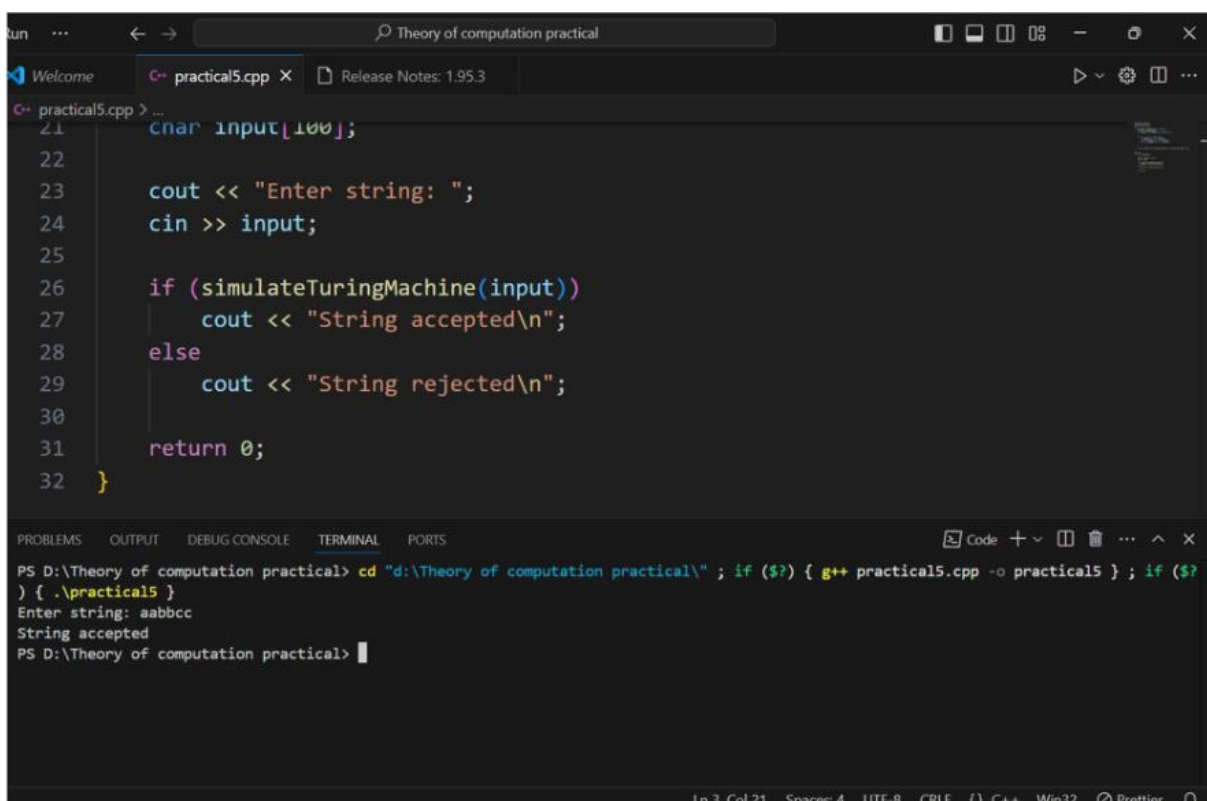
```
#include <iostream>
#include <string.h>
using namespace std;
int simulateTuringMachine(char* input) {
    int len = strlen(input);
    int a_count = 0, b_count = 0, c_count = 0;
    for (int i = 0; i < len; i++) {
        if (input[i] == 'a') a_count++;
        else if (input[i] == 'b') b_count++;
        else if (input[i] == 'c') c_count++;
    }
    return (a_count == b_count && b_count == c_count && a_count > 0);
}
main() {
    char input[100];
    cout << "Enter string: ";
    cin >> input;
    if (simulateTuringMachine(input))
        cout << "String accepted\n";
    else
        cout << "String rejected\n";
    return 0;
}
```

Output:



```
practical5.cpp
21 char input[100];
22
23 cout << "Enter string: ";
24 cin >> input;
25
26 if (simulateTuringMachine(input))
27     cout << "String accepted\n";
28 else
29     cout << "String rejected\n";
30
31 return 0;
32 }
```

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter string: aabbcc
String rejected
PS D:\Theory of computation practical>
```



```
practical5.cpp
21 char input[100];
22
23 cout << "Enter string: ";
24 cin >> input;
25
26 if (simulateTuringMachine(input))
27     cout << "String accepted\n";
28 else
29     cout << "String rejected\n";
30
31 return 0;
32 }
```

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter string: aabbcc
String accepted
PS D:\Theory of computation practical>
```

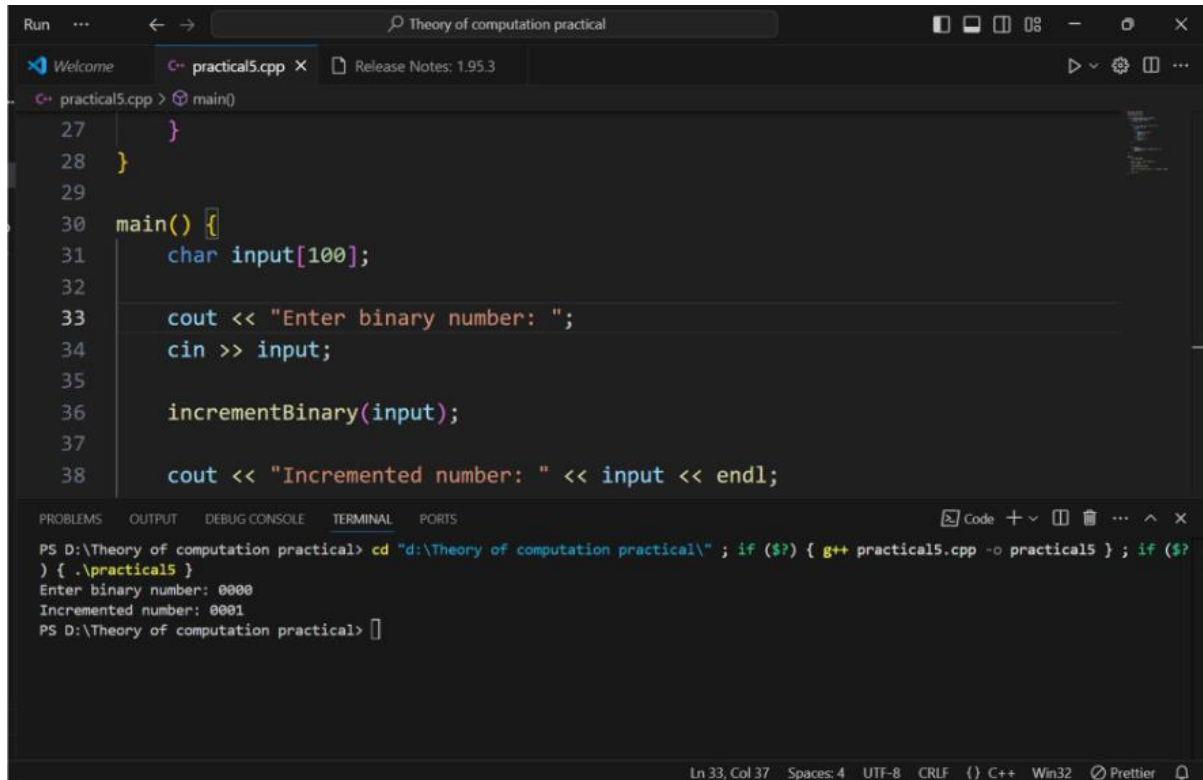
10) Design and simulate a Turing Machine that accepts the language $a^n b^n c^n$ where $n > 0$.

```
#include <iostream>
#include <string.h>
using namespace std;
void incrementBinary(char* input) {
    int len = strlen(input);
    int carry = 1;
    for (int i = len - 1; i >= 0; i--) {
        if (carry == 0)
            break;
        if (input[i] == '0') {
            input[i] = '1';
            carry = 0;
        } else {
            input[i] = '0';
            carry = 1;
        }
    }
    if (carry) {
        memmove(input + 1, input, len + 1);
        input[0] = '1';
    }
}
main() {
    char input[100];
    cout << "Enter binary number: ";
    cin >> input;
    incrementBinary(input);
    cout << "Incremented number: " << input << endl;
```

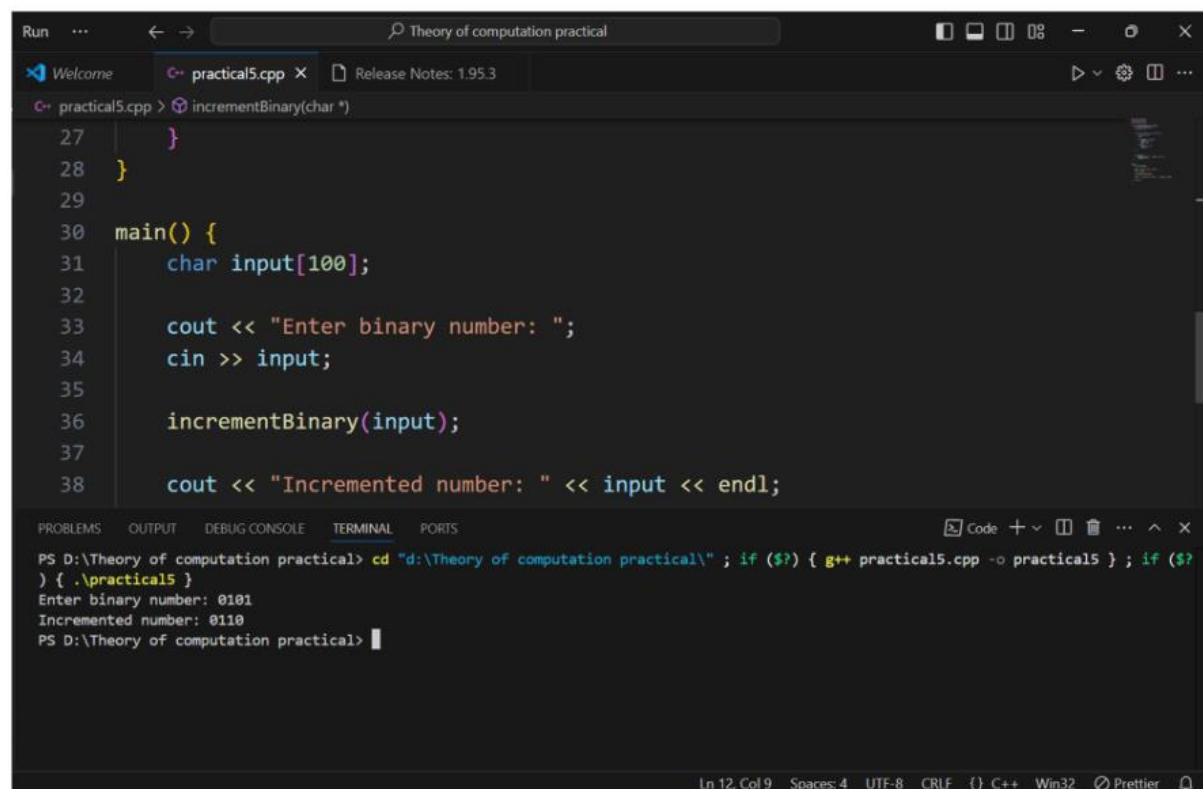
return 0;

}

Output:



```
Run ... Theory of computation practical
practical5.cpp x Release Notes: 1.95.3
practical5.cpp > main()
27     }
28 }
29
30 main() {
31     char input[100];
32
33     cout << "Enter binary number: ";
34     cin >> input;
35
36     incrementBinary(input);
37
38     cout << "Incremented number: " << input << endl;
39 }
40
41 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter binary number: 0000
Incremented number: 0001
PS D:\Theory of computation practical>
```



```
Run ... Theory of computation practical
practical5.cpp x Release Notes: 1.95.3
practical5.cpp > incrementBinary(char *)
27     }
28 }
29
30 main() {
31     char input[100];
32
33     cout << "Enter binary number: ";
34     cin >> input;
35
36     incrementBinary(input);
37
38     cout << "Incremented number: " << input << endl;
39 }
40
41 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?) { .\practical5 }
Enter binary number: 0101
Incremented number: 0110
PS D:\Theory of computation practical>
```