PROGRAM

Name : Ankit Kumar

Course : B Sc (H) Computer Science

Roll no : 16005

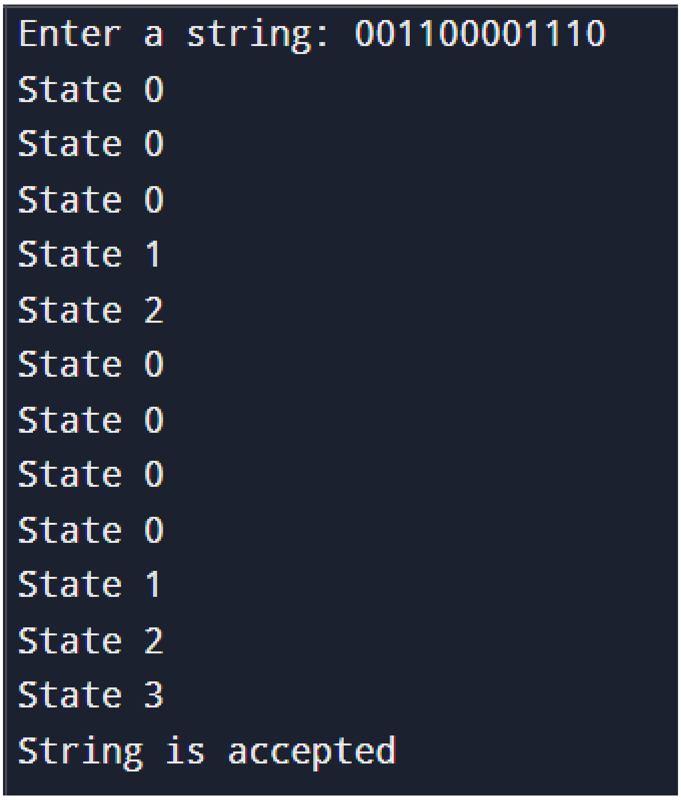
Subject: THEORY OF COMPUTATION

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
}
```



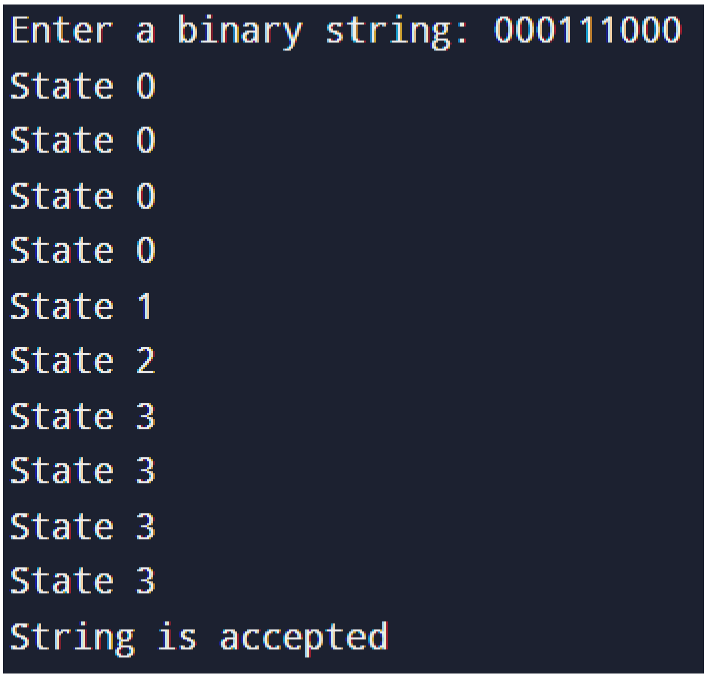
```
Enter a string: 001100001110
State 0
State 0
State 0
State 1
State 2
State 0
State 0
State 0
State 0
State 1
State 2
State 3
String is accepted
```

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
```

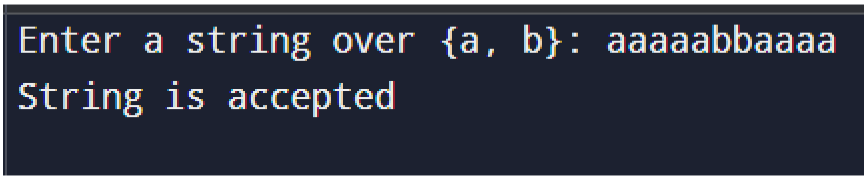
Q3) Design a Finite Automata (FA) that accepts language L1, over S={a, b}, comprising of all strings (of length 4 or more) having first two characters same as the last two. Write a program to simulate this FA.

```
#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:

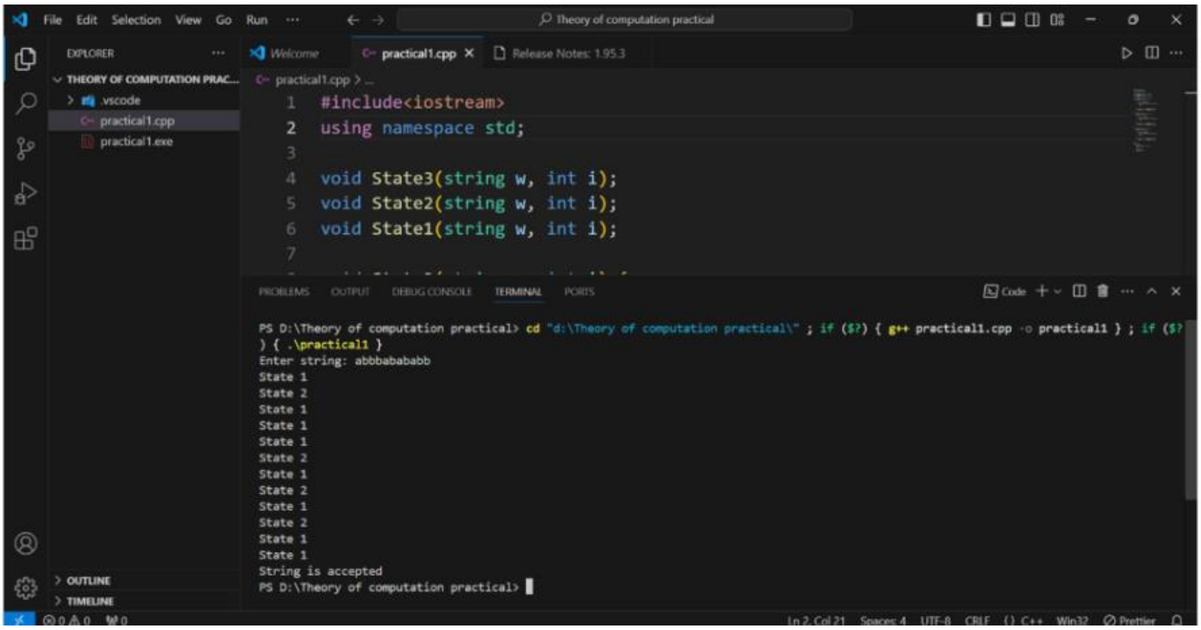


```
Enter a string over {a, b}: aaaaabbaaaa
String is accepted
```

Q4) 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.

```
#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:



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

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

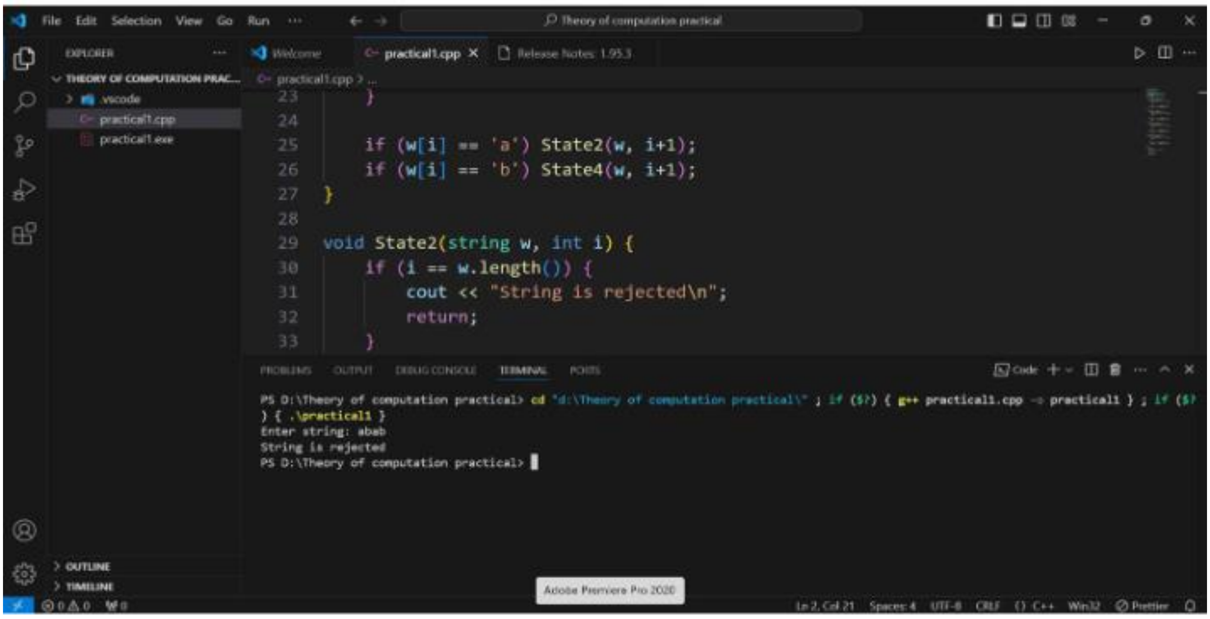
```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical1.cpp -o practical1 } ; if ($?
) { .\practical1 }
Enter string: abbbabababb
State 1
State 2
State 1
State 1
State 1
State 2
State 1
State 2
State 1
State 2
State 1
State 1
String is accepted
PS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



```
    }

    if (w[i] == 'a') State2(w, i+1);
    if (w[i] == 'b') State4(w, i+1);
}

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

```
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>
```

File Edit Selection View Go Run ···
Theory of computation practical
EXPLORER
THEORY OF COMPUTATION PRAC...
.vscode
practical1.cpp
practical1.exe
Welcome
practical1.cpp
Release Notes: 1.95.3
practical1.cpp > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
OUTLINE
TIMELINE
Adobe Premiere Pro 2020
Ln 2, Col 21 Spaces: 4 UTF-8 CRLF {} C++ Win32 Prettier

Q6) Write a program to simulate an FA that accepts

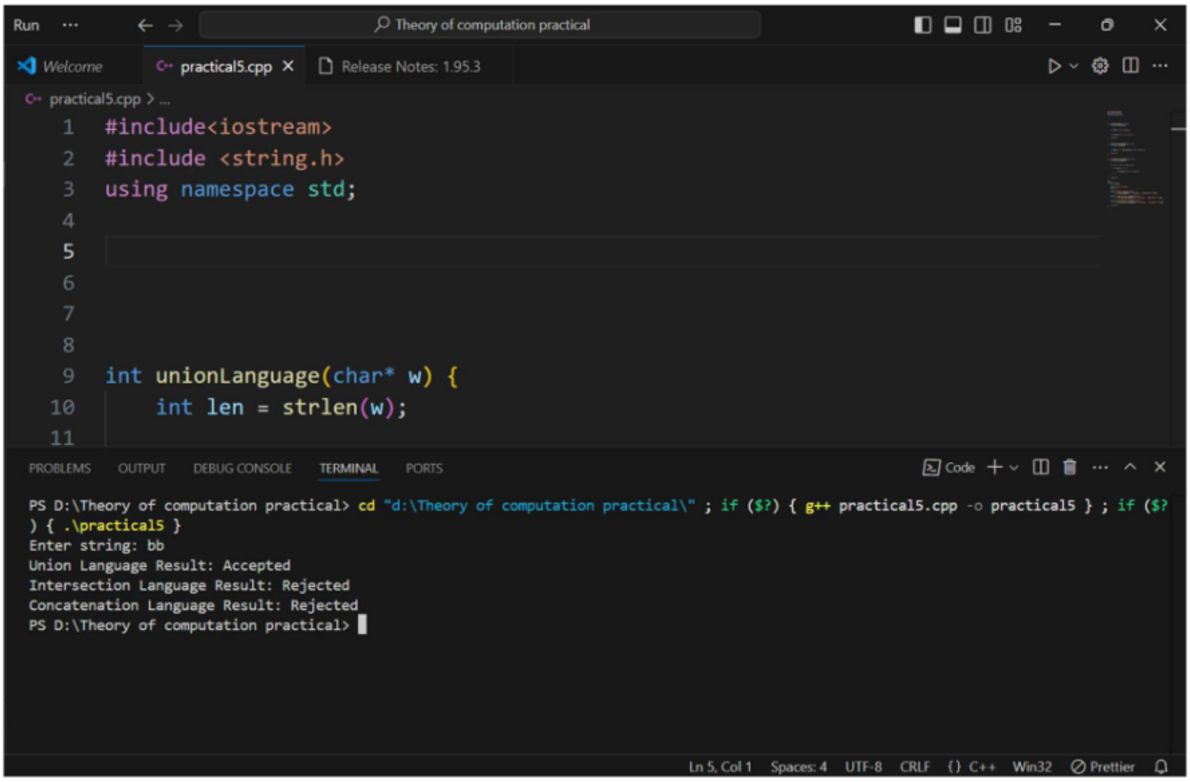
a. Union of the languages L1 and L2

b. Intersection of the languages L1 and L2

c. Language L1 L2 (concatenation).

```
#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:



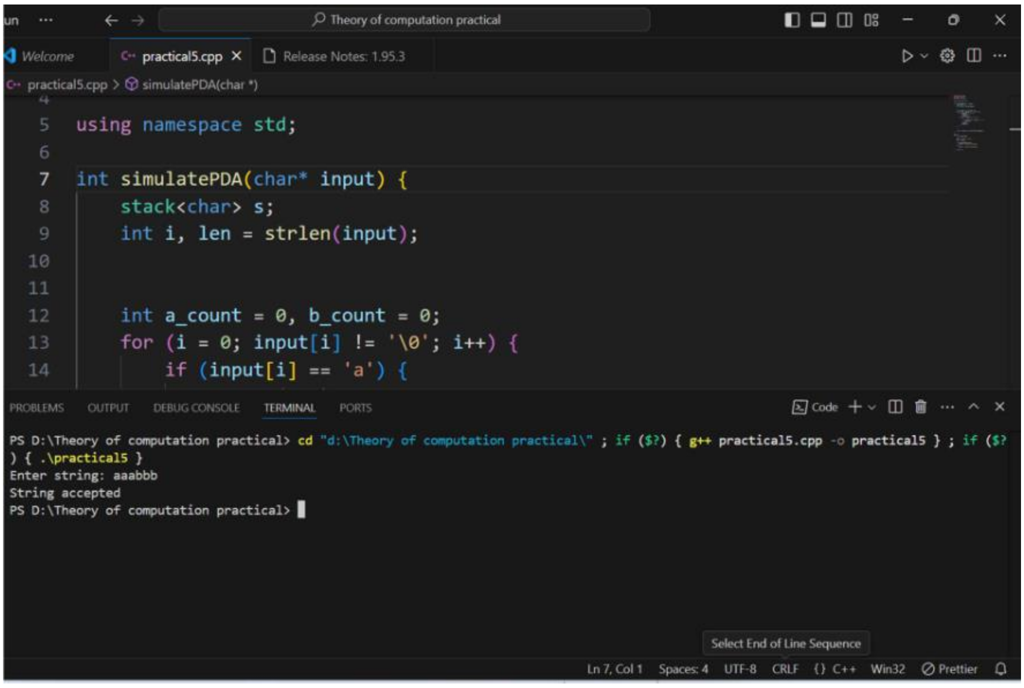
```cpp
#include<iostream>
#include <string.h>
using namespace std;




int unionLanguage(char* w) {
    int len = strlen(w);

```

```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?
) { .\practical5 }
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^nb^n 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:



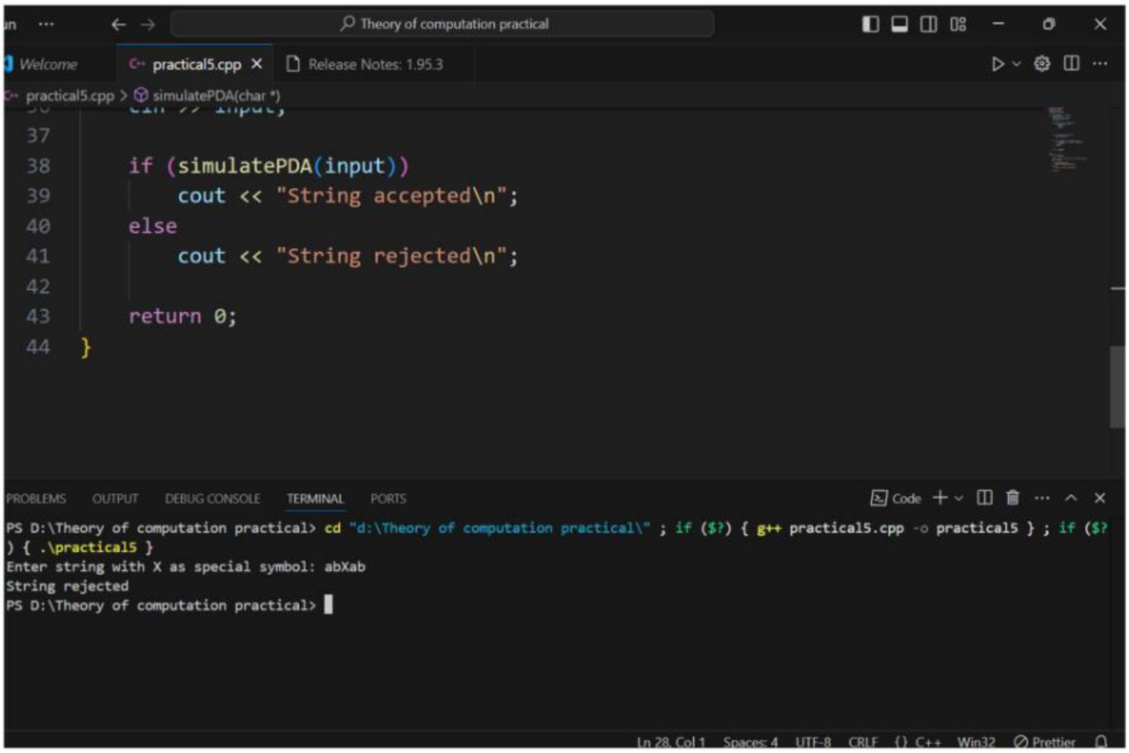
```
using namespace std;

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') {
```

```
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| w is any string over S={a, b} and w^r is reverse of that string and X is a special symbol }.

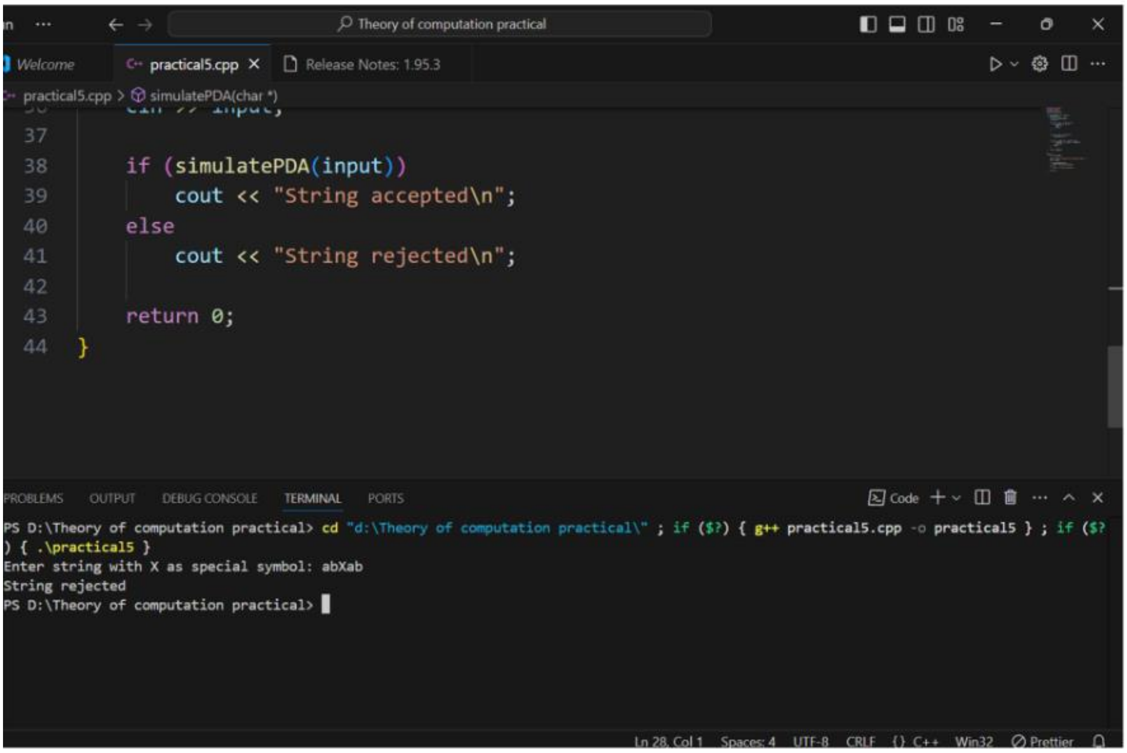
```
#include<iostream>

#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:



```
    cin >> input;

    if (simulatePDA(input))
        cout << "String accepted\n";
    else
        cout << "String rejected\n";

    return 0;
}
```

```
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> 
```



Theory of computation practical
Welcome
practical5.cpp
Release Notes: 1.95.3
practical5.cpp > simulatePDA(char *)

```
    cin >> input;

    if (simulatePDA(input))
        cout << "String accepted\n";
    else
        cout << "String rejected\n";

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

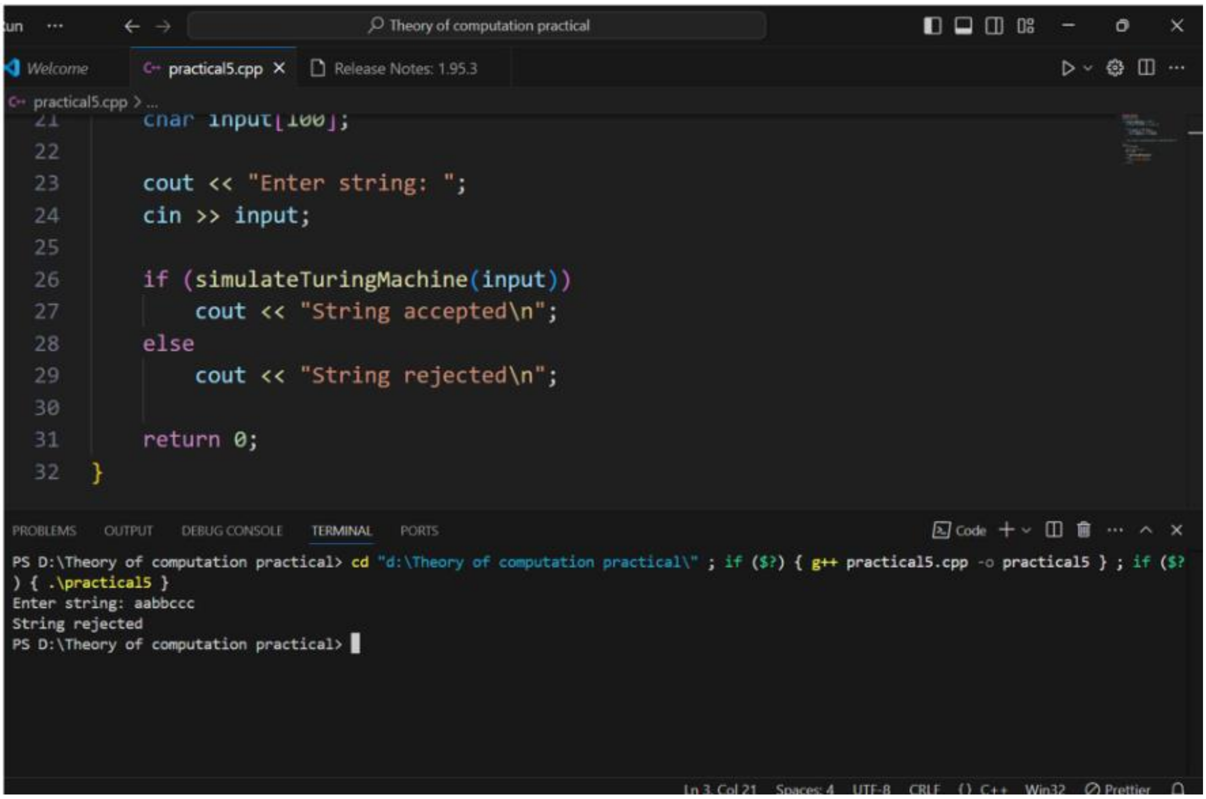
```
PS D:\Theory of computation practical> cd "d:\Theory of computation practical\" ; if ($?) { g++ practical5.cpp -o practical5 } ; if ($?
) { .\practical5 }
Enter string with X as special symbol: abXab
String rejected
PS D:\Theory of computation practical> 
```

Ln 28, Col 1 Spaces: 4 UTF-8 CRLF {} C++ Win32 Prettier

Q9) Design and simulate a Turing Machine that accepts the language a^nb^nc^n where n >0.

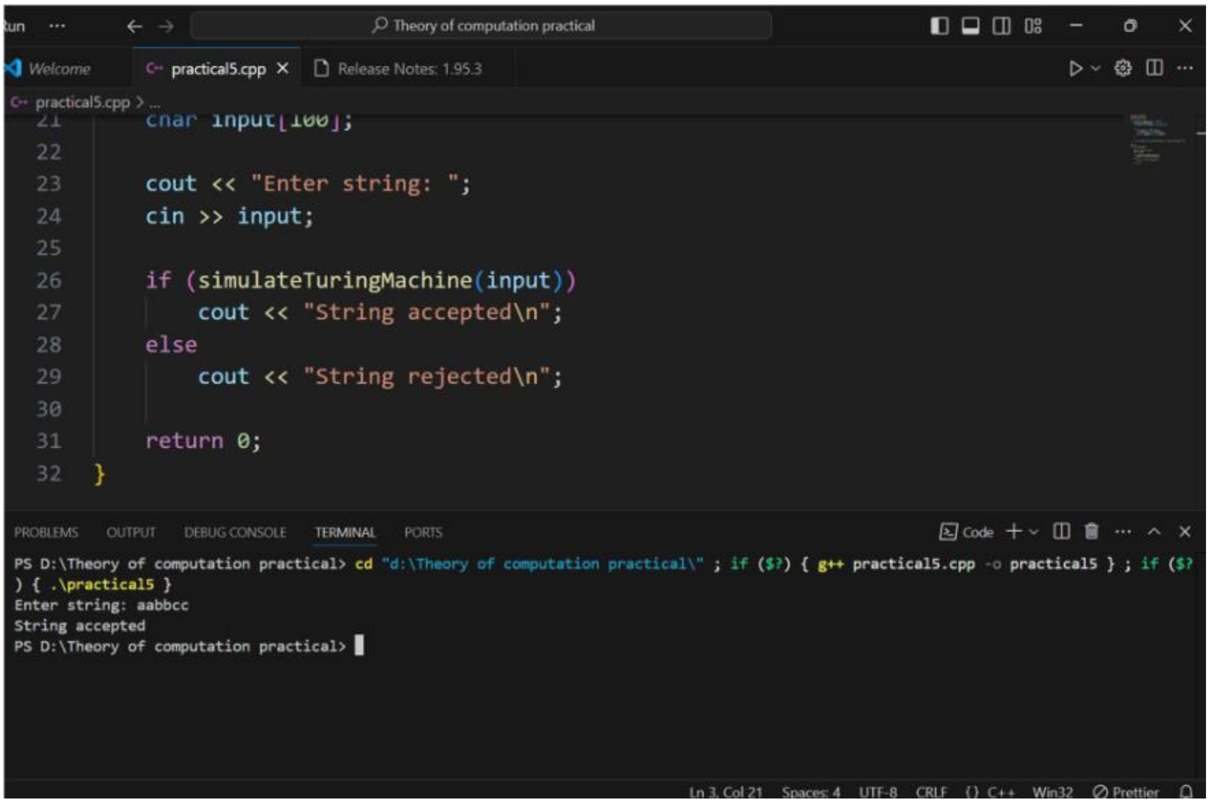
```
#include <iostream>
#include <string.h>
using namespace std;
int simulateTuringMachine(char* input) {
int len = strlen(input);
int a_count = 0, b_count = 0, c_count = 0;
for (int i = 0; i < len; i++) {
if (input[i] == 'a') a_count++;
else if (input[i] == 'b') b_count++;
else if (input[i] == 'c') 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:


Theory of computation practical
Welcome
practical5.cpp
Release Notes: 1.95.3
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Ln 3, Col 21 Spaces: 4 UTF-8 CRLF {} C++ Win32 Prettier

```cpp
    char input[100];

    cout << "Enter string: ";
    cin >> input;

    if (simulateTuringMachine(input))
        cout << "String accepted\n";
    else
        cout << "String rejected\n";

    return 0;
}
```

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


Theory of computation practical
Welcome
practical5.cpp
Release Notes: 1.95.3
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Ln 3, Col 21 Spaces: 4 UTF-8 CRLF {} C++ Win32 Prettier

```cpp
    char input[100];

    cout << "Enter string: ";
    cin >> input;

    if (simulateTuringMachine(input))
        cout << "String accepted\n";
    else
        cout << "String rejected\n";

    return 0;
}
```

```
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> 
```

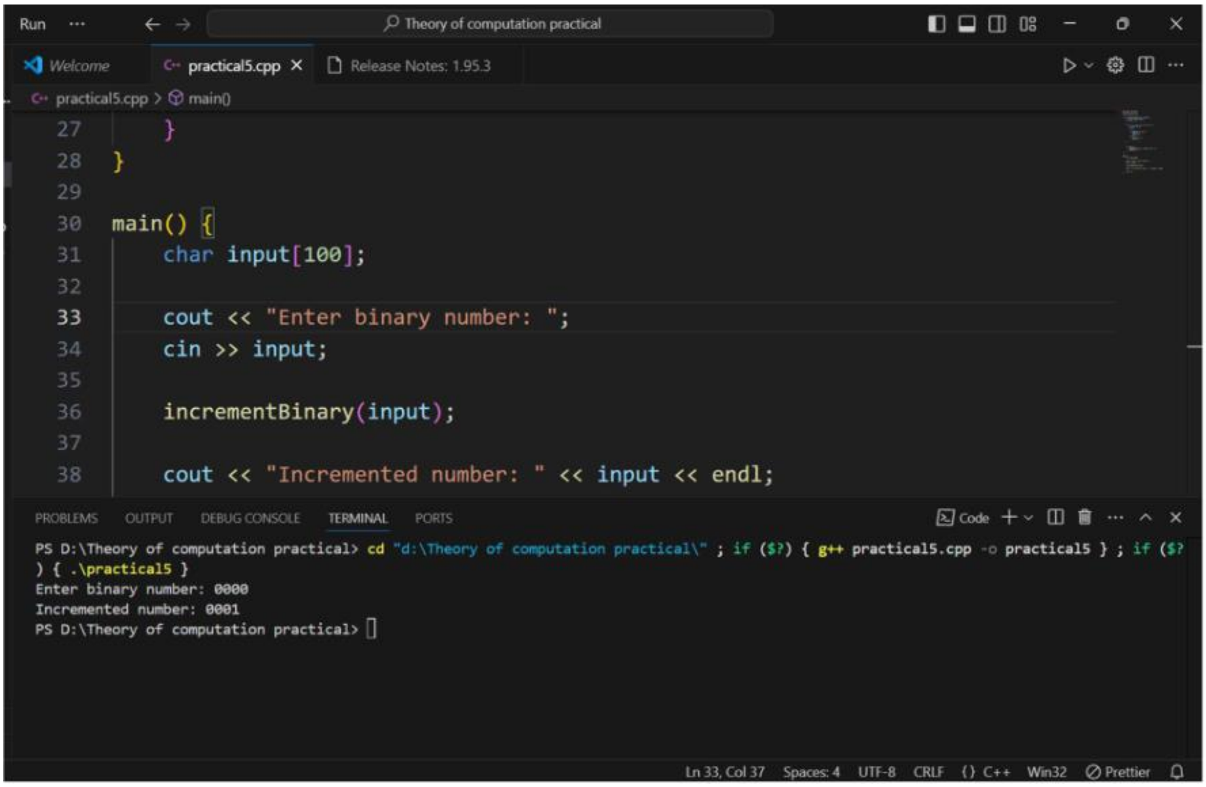
10) Design and simulate a Turing Machine that accepts the language a^nb^nc^n where n >0.

```
#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;

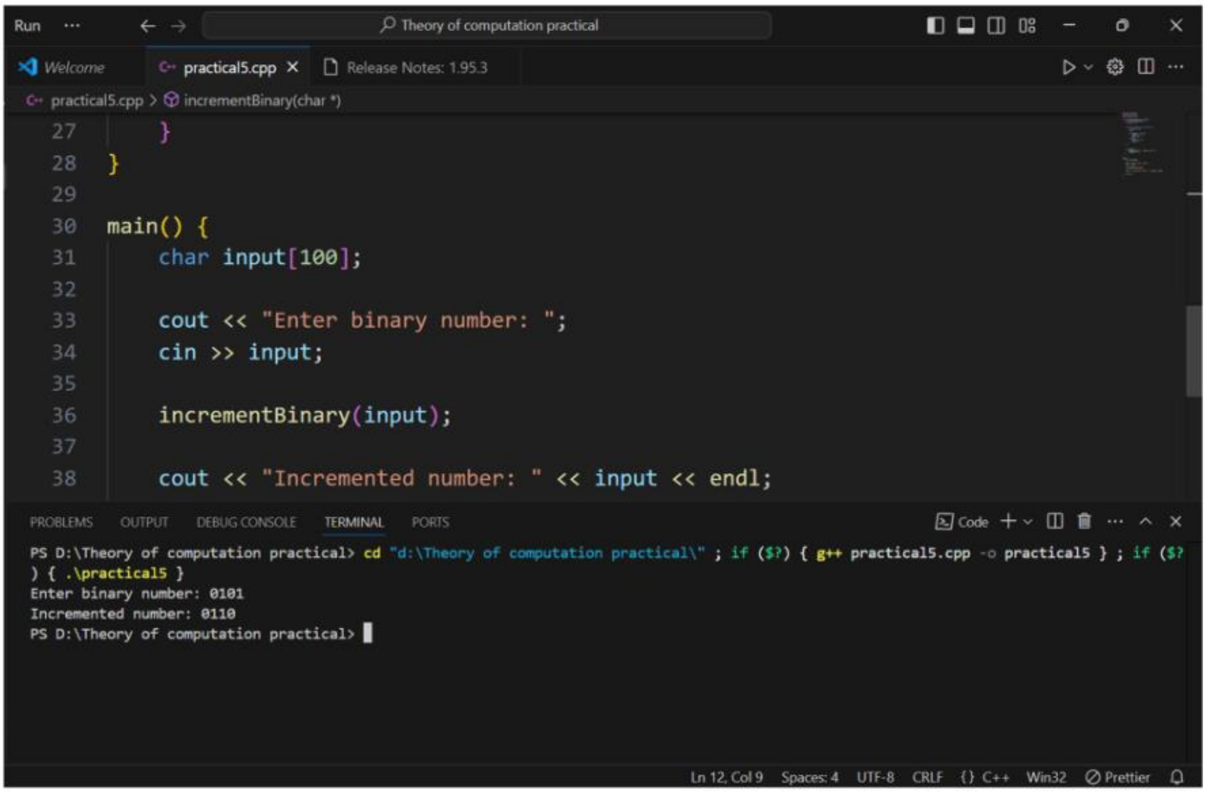
}

Output:



```
    }
}

main() {
    char input[100];

    cout << "Enter binary number: ";
    cin >> input;

    incrementBinary(input);

    cout << "Incremented number: " << input << endl;
```

```
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>
```



```
    }
}

main() {
    char input[100];

    cout << "Enter binary number: ";
    cin >> input;

    incrementBinary(input);

    cout << "Incremented number: " << input << endl;
```

```
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>
```