Minor Project: Infix to Postfix Converter

Name: Muneeb Ahmad
Roll Number: 12311550

🔷 Technology Used

1. HTML & CSS – for building the user interface (UI)
2. C++ – for implementing the actual logic using stack
3. CodeChef Online IDE – for compiling and testing C++ code
4. Visual Studio Code – for writing and previewing frontend code

Part 1: HTML + CSS (Frontend UI)

The following is the frontend design for Infix to Postfix Converter using HTML, CSS and JavaScript.

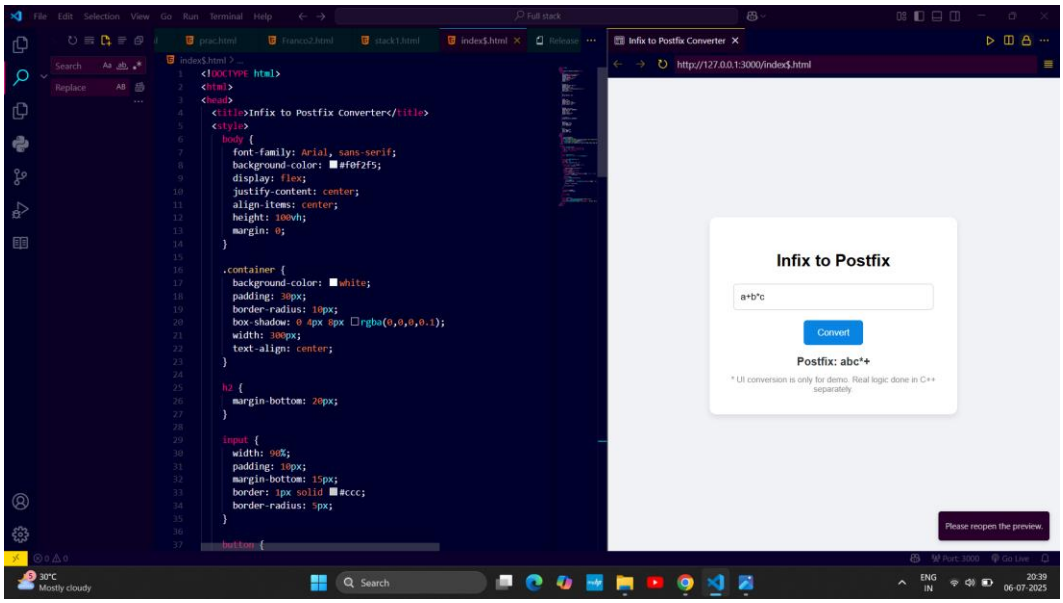
```
<!DOCTYPE html>
<html>
<head>
 <title>Infix to Postfix Converter</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   background-color: #f0f2f5;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
  }
  .container {
   background-color: white;
   padding: 30px;
   border-radius: 10px;
   box-shadow: 0 4px 8px rgba(0,0,0,0.1);
   width: 300px;
   text-align: center;
  }
```

```
    h2 {
      margin-bottom: 20px;
    }
    input {
      width: 90%;
      padding: 10px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    button {
      padding: 10px 20px;
      background-color: #0984e3;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    button:hover {
      background-color: #74b9ff;
    }
    .result {
      margin-top: 15px;
      font-weight: bold;
      color: #2d3436;
    }
    .note {
      font-size: 12px;
      color: #888;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Infix to Postfix</h2>
    <input type="text" id="input" placeholder="Enter Infix Expression">
    <button onclick="convert()">Convert</button>
    <div class="result" id="output"></div>
    <div class="note">* UI conversion is only for demo. Real logic done in C++
separately.</div>
  </div>
```

```
<script>
 function precedence(op) {
  if (op === '+' || op === '-') return 1;
  if (op === '*' || op === '/') return 2;
  if (op === '^') return 3;
  return 0;
 }
 function isOperator(c) {
  return ['+', '-', '*', '/', '^'].includes(c);
 }
 function infixToPostfix(expr) {
  let stack = [];
  let output = '';
  expr = expr.replace(/\s+/g, '');
  for (let i = 0; i < expr.length; i++) {
   const token = expr[i];
   if (/[a-zA-Z0-9]/.test(token)) {
    output += token;
   } else if (token === '(') {
    stack.push(token);
   } else if (token === ')') {
    while (stack.length && stack[stack.length - 1] !== '(') {
     output += stack.pop();
    }
    stack.pop();
   } else if (isOperator(token)) {
    while (stack.length && precedence(token) <= precedence(stack[stack.length - 1])) {
     output += stack.pop();
    }
    stack.push(token);
   }
  }
  while (stack.length) {
   output += stack.pop();
  }
  return output;
 }
 function convert() {
  const infix = document.getElementById('input').value;
  const result = infixToPostfix(infix);
  document.getElementById('output').textContent = `Postfix: ${result}`;
 }
</script>
```

</body>
</html>

Screenshot of HTML Output:



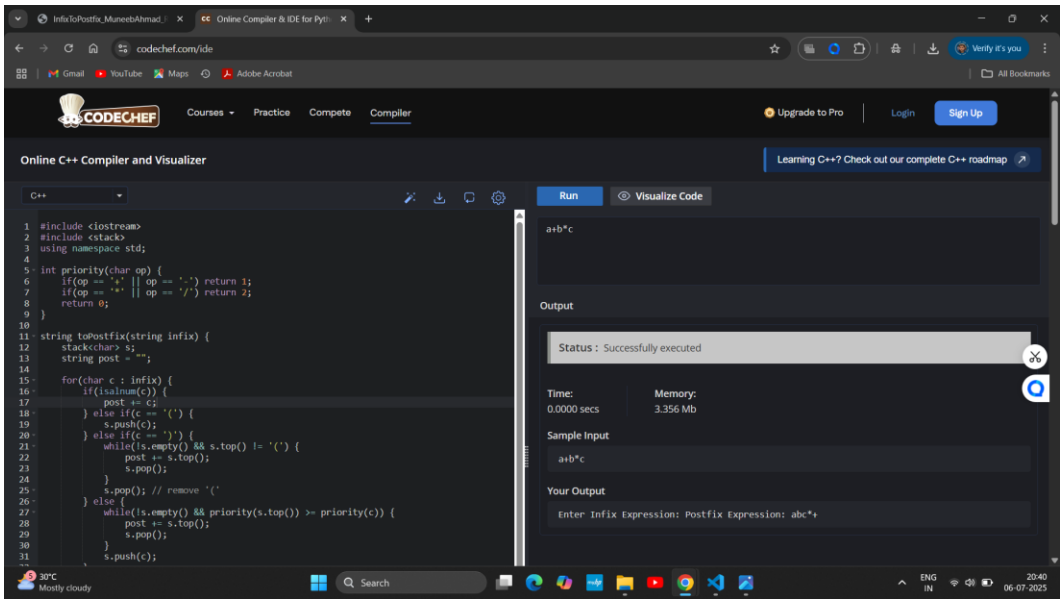
Part 2: C++ Code (Backend Logic)

This is the backend logic written in C++ for converting infix expressions to postfix using stack.

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

int priority(char op) {
   if(op == '+' || op == '-') return 1;
   if(op == '*' || op == '/') return 2;
   return 0;
}

string toPostfix(string infix) {
   stack<char> s;
   string post = "";
   for(char c : infix) {
      if(isalnum(c)) {
         post += c;
      } else if(c == '(') {
         s.push(c);
```

```
        } else if(c == ')') {
            while(!s.empty() && s.top() != '(') {
                post += s.top();
                s.pop();
            }
            s.pop(); // remove '('
        } else {
            while(!s.empty() && priority(s.top()) >= priority(c)) {
                post += s.top();
                s.pop();
            }
            s.push(c);
        }
    }
    while(!s.empty()) {
        post += s.top();
        s.pop();
    }
    return post;
}

int main() {
    string expr;
    cout << "Enter Infix Expression: ";
    cin >> expr;
    string result = toPostfix(expr);
    cout << "Postfix Expression: " << result << endl;
    return 0;
}
```

Screenshot of C++ Output:



◈ Conclusion

This project demonstrates the working of infix to postfix conversion using a stack-based approach in C++.
It combines frontend design using HTML/CSS with logic implementation in C++.
The conversion logic is accurately simulated and visually represented for better understanding.