

```
1  #include <iostream>
2  #include <stack>
3  #include <string>
4
5  int precedence(char op) {
6      if (op == '+' || op == '-')
7          return 1;
8      if (op == '*' || op == '/')
9          return 2;
10     return 0;
11 }
12
13 std::string infixToPostfix(const std::string& infixExpression) {
14     std::string postfix;
15     std::stack<char> operatorStack;
16
17     for (char ch : infixExpression) {
18         if (isdigit(ch)) {
19             postfix += ch;
20         } else if (ch == '(') {
21             operatorStack.push(ch);
22         } else if (ch == ')') {
23             while (!operatorStack.empty() && operatorStack.top() != '(') {
24                 postfix += operatorStack.top();
25                 operatorStack.pop();
26             }
27             operatorStack.pop();
28         } else {
29             while (!operatorStack.empty() && precedence(operatorStack.top()
30                 >= precedence(ch)) {
31                 postfix += operatorStack.top();
32                 operatorStack.pop();
33             }
34             operatorStack.push(ch);
35         }
36     }
37
38     while (!operatorStack.empty()) {
39         postfix += operatorStack.top();
40         operatorStack.pop();
41     }
42
43     return postfix;
44 }
```

```
49
50 int main() {
51     std::string infixExpression;
52     std::cout << "Enter an infix expression: ";
53     std::getline(std::cin, infixExpression);
54     std::string postfixExpression = infixToPostfix(infixExpression);
55     std::cout << "Postfix expression: " << postfixExpression << std::endl;
56
57     return 0;
58 }
59
```