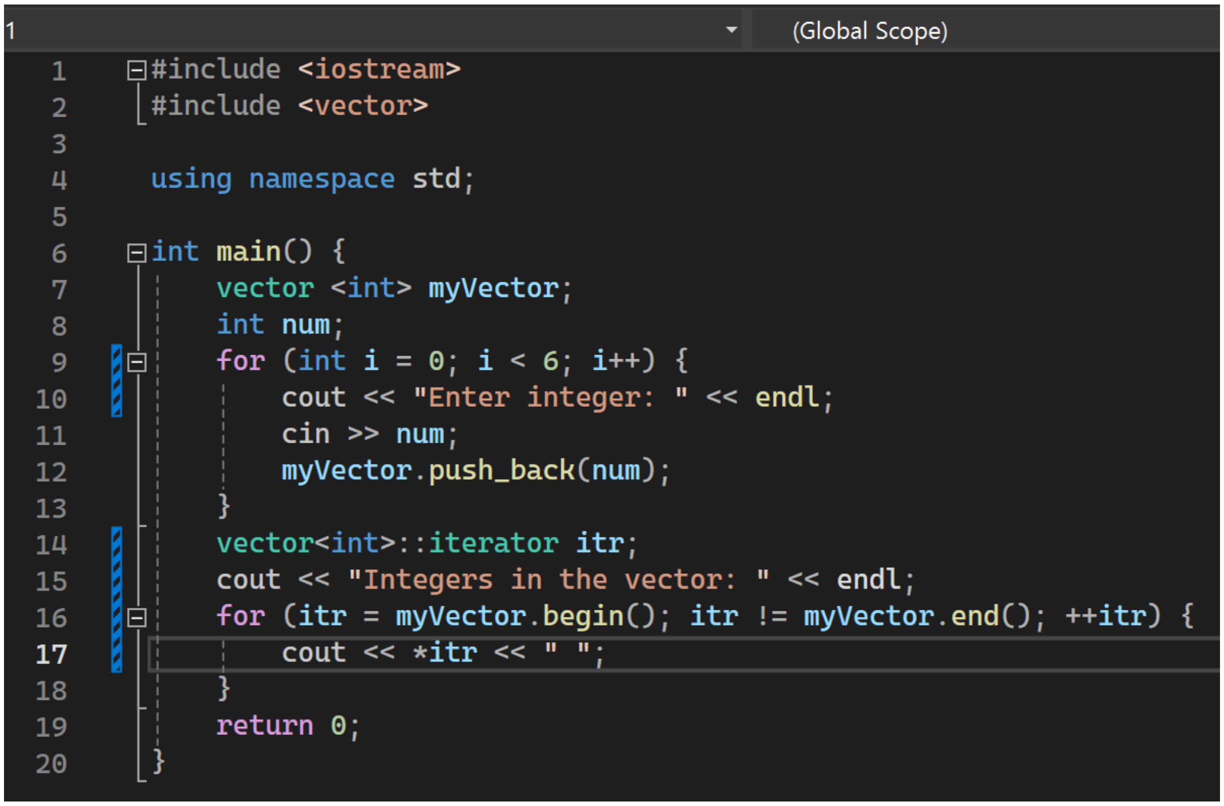
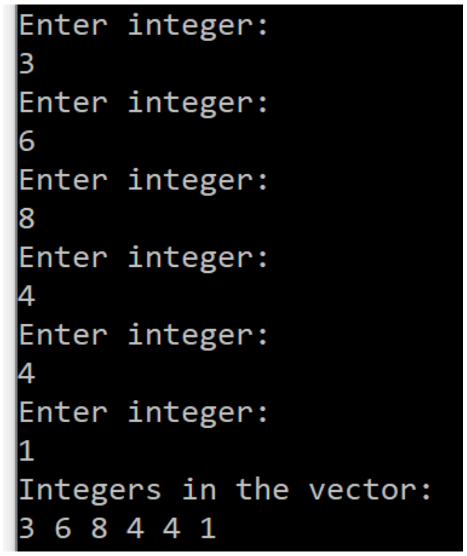
Q1
Code



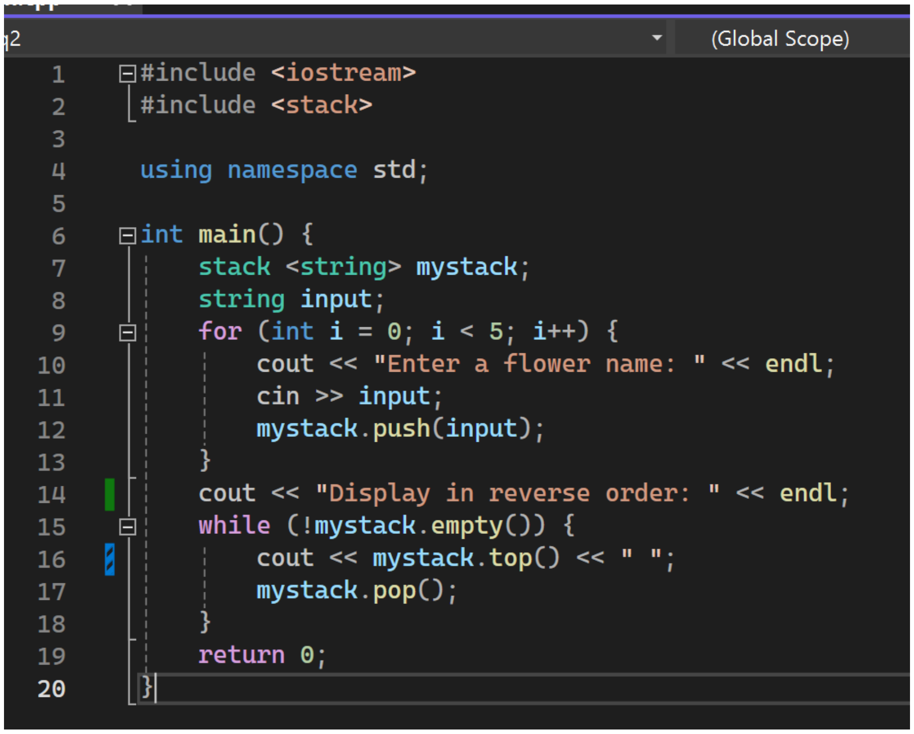
```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector <int> myVector;
    int num;
    for (int i = 0; i < 6; i++) {
        cout << "Enter integer: " << endl;
        cin >> num;
        myVector.push_back(num);
    }
    vector<int>::iterator itr;
    cout << "Integers in the vector: " << endl;
    for (itr = myVector.begin(); itr != myVector.end(); ++itr) {
        cout << *itr << " ";
    }
    return 0;
}
```

Output



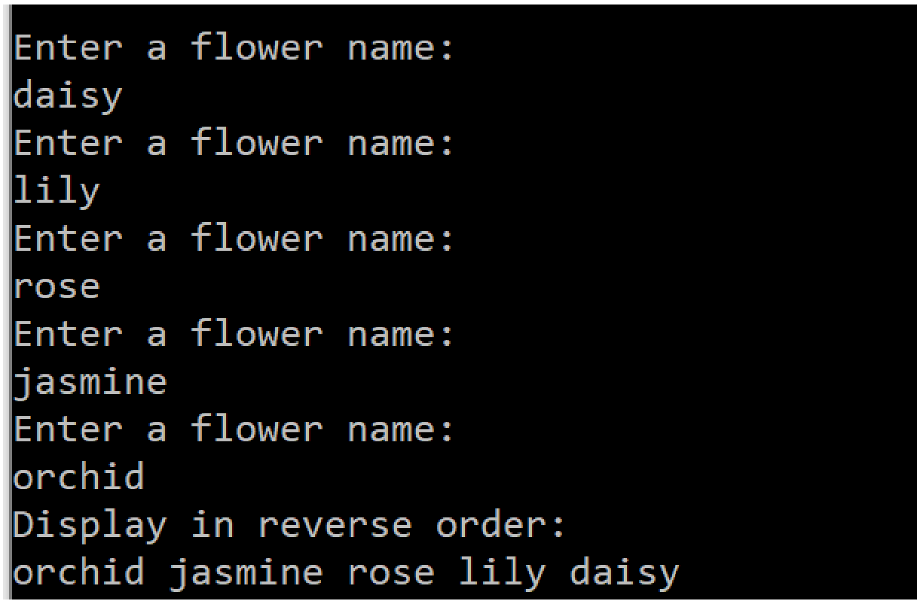
```
Enter integer:
3
Enter integer:
6
Enter integer:
8
Enter integer:
4
Enter integer:
4
Enter integer:
1
Integers in the vector:
3 6 8 4 4 1
```

Q2
Code



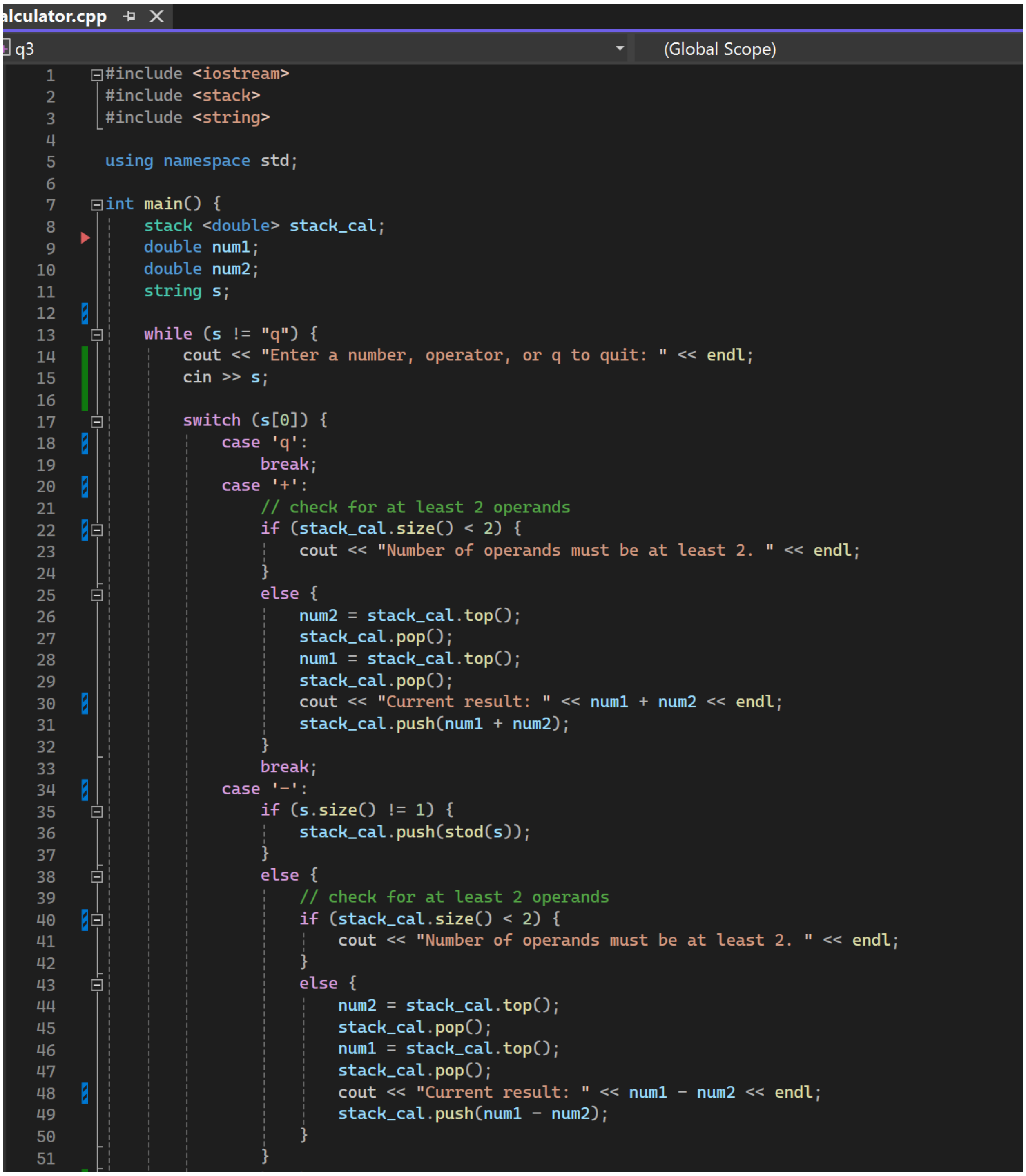
```cpp
#include <iostream>
#include <stack>

using namespace std;

int main() {
    stack <string> mystack;
    string input;
    for (int i = 0; i < 5; i++) {
        cout << "Enter a flower name: " << endl;
        cin >> input;
        mystack.push(input);
    }
    cout << "Display in reverse order: " << endl;
    while (!mystack.empty()) {
        cout << mystack.top() << " ";
        mystack.pop();
    }
    return 0;
}
```

Output



```
Enter a flower name:
daisy
Enter a flower name:
lily
Enter a flower name:
rose
Enter a flower name:
jasmine
Enter a flower name:
orchid
Display in reverse order:
orchid jasmine rose lily daisy
```

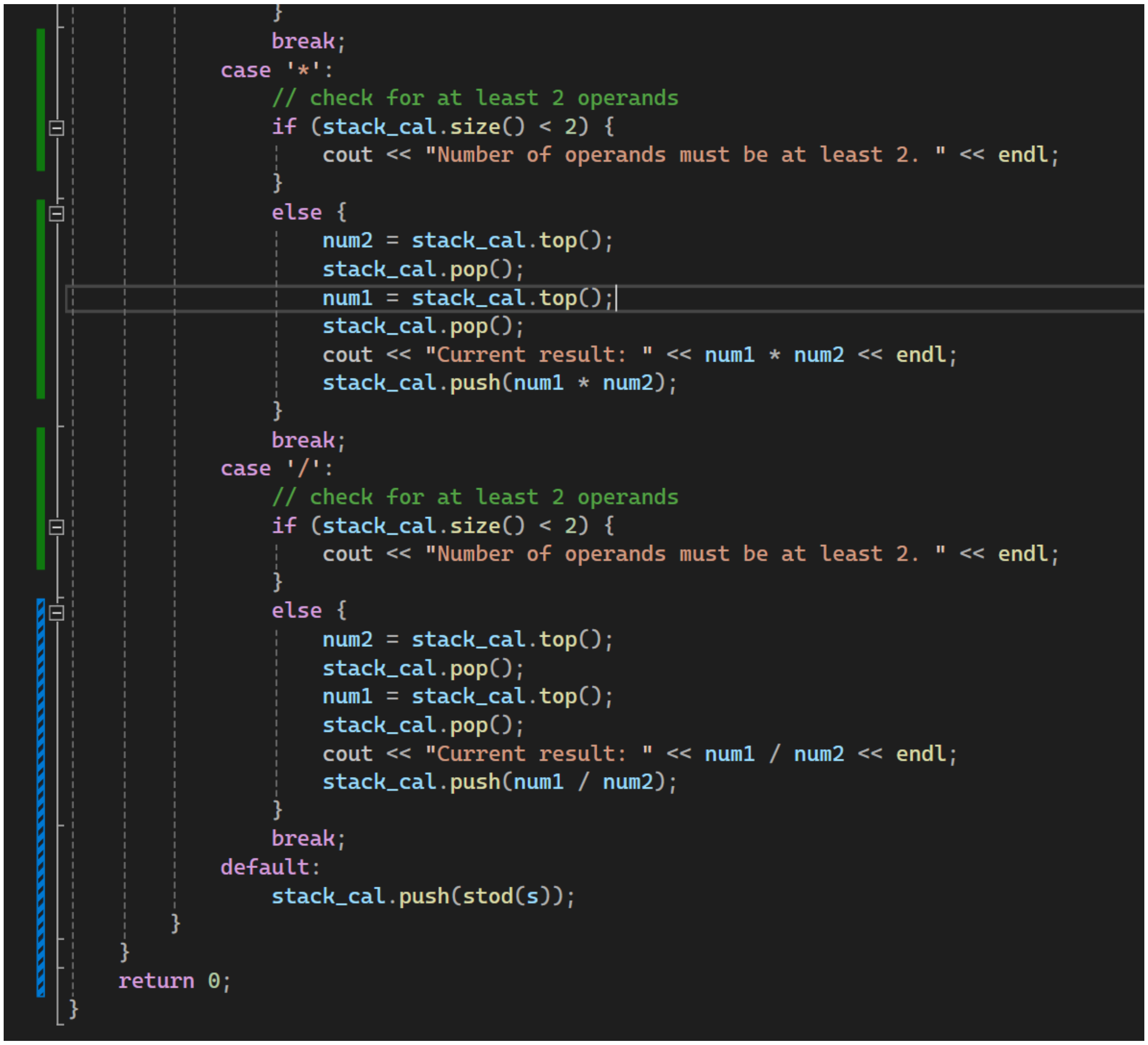
Q3
Code



alculator.cpp

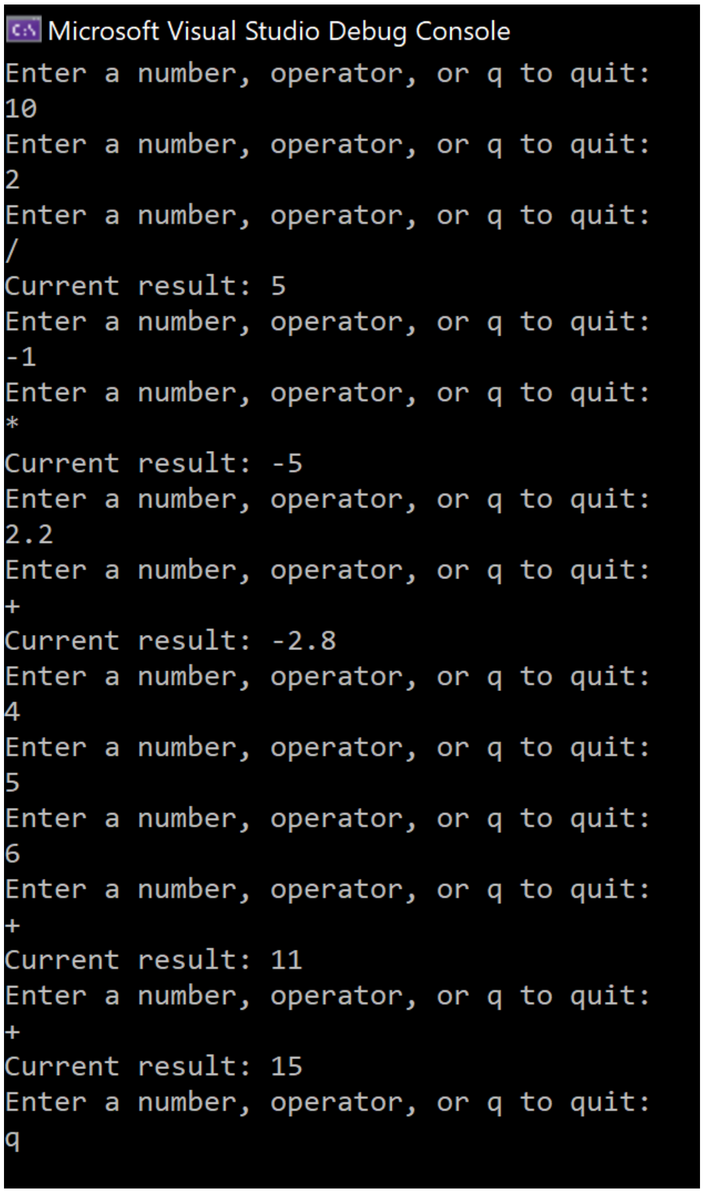
q3 (Global Scope)

```cpp
#include <iostream>
#include <stack>
#include <string>

using namespace std;

int main() {
    stack <double> stack_cal;
    double num1;
    double num2;
    string s;

    while (s != "q") {
        cout << "Enter a number, operator, or q to quit: " << endl;
        cin >> s;

        switch (s[0]) {
            case 'q':
                break;
            case '+':
                // check for at least 2 operands
                if (stack_cal.size() < 2) {
                    cout << "Number of operands must be at least 2. " << endl;
                }
                else {
                    num2 = stack_cal.top();
                    stack_cal.pop();
                    num1 = stack_cal.top();
                    stack_cal.pop();
                    cout << "Current result: " << num1 + num2 << endl;
                    stack_cal.push(num1 + num2);
                }
                break;
            case '-':
                if (s.size() != 1) {
                    stack_cal.push(stod(s));
                }
                else {
                    // check for at least 2 operands
                    if (stack_cal.size() < 2) {
                        cout << "Number of operands must be at least 2. " << endl;
                    }
                    else {
                        num2 = stack_cal.top();
                        stack_cal.pop();
                        num1 = stack_cal.top();
                        stack_cal.pop();
                        cout << "Current result: " << num1 - num2 << endl;
                        stack_cal.push(num1 - num2);
                    }
                }
```



```
                    }
                    break;
                case '*':
                    // check for at least 2 operands
                    if (stack_cal.size() < 2) {
                        cout << "Number of operands must be at least 2. " << endl;
                    }
                    else {
                        num2 = stack_cal.top();
                        stack_cal.pop();
                        num1 = stack_cal.top();
                        stack_cal.pop();
                        cout << "Current result: " << num1 * num2 << endl;
                        stack_cal.push(num1 * num2);
                    }
                    break;
                case '/':
                    // check for at least 2 operands
                    if (stack_cal.size() < 2) {
                        cout << "Number of operands must be at least 2. " << endl;
                    }
                    else {
                        num2 = stack_cal.top();
                        stack_cal.pop();
                        num1 = stack_cal.top();
                        stack_cal.pop();
                        cout << "Current result: " << num1 / num2 << endl;
                        stack_cal.push(num1 / num2);
                    }
                    break;
                default:
                    stack_cal.push(stod(s));
            }
        }
    return 0;
}
```

Output


Microsoft Visual Studio Debug Console

```
Enter a number, operator, or q to quit:
10
Enter a number, operator, or q to quit:
2
Enter a number, operator, or q to quit:
/
Current result: 5
Enter a number, operator, or q to quit:
-1
Enter a number, operator, or q to quit:
*
Current result: -5
Enter a number, operator, or q to quit:
2.2
Enter a number, operator, or q to quit:
+
Current result: -2.8
Enter a number, operator, or q to quit:
4
Enter a number, operator, or q to quit:
5
Enter a number, operator, or q to quit:
6
Enter a number, operator, or q to quit:
+
Current result: 11
Enter a number, operator, or q to quit:
+
Current result: 15
Enter a number, operator, or q to quit:
q
```