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

using namespace std;

// Function to evaluate a postfix expression
int evaluatePostfix(const string& expression)
{
    stack<int> s;
    int values[26] = {0}; // Array to store values for variables (A-Z)

    for (char ch : expression)
    {
        if (isdigit(ch))
        {
            // If it's a digit, push its integer value onto the stack
            s.push(ch - '0');
        }
        else if (isalpha(ch))
        {
            // If it's a letter, ask the user for its value
            if (values[ch - 'A'] == 0)   // Assume the variable is not initialized
            {
                cout << "Enter value for " << ch <<"(single digit): ";
                cin >> values[ch - 'A'];
            }
            s.push(values[ch - 'A']);
        }
        else
        {
            // It's an operator, pop two operands from the stack and apply the operator
            int operand2 = s.top();
            s.pop();
            int operand1 = s.top();
            s.pop();
            int result = 0;

            switch (ch)
            {
            case '+':
                result = operand1 + operand2;
                break;
            case '-':
                result = operand1 - operand2;
                break;
            case '*':
                result = operand1 * operand2;
                break;
            case '/':
                result = operand1 / operand2;
                break;
            case '^':
                result = 1;
                for(int i = 0; i < operand2; i++) result *= operand1;
                break;
            default:
                cout << "Invalid operator!" << endl;
                return -1;
            }

            s.push(result);
        }
    }

    return s.top(); // The final result
```

```
}

int main()
{
    string expression;
    cout << "Enter the postfix expression: ";
    cin >> expression;

    int result = evaluatePostfix(expression);

    cout << "The result of the expression is: " << result << endl;

    return 0;
}
```