```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

int isOperator(char ch) {
   return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^');
}

int precedence(char op) {
   switch(op) {
      case '^':
         return 3;
      case '*':
      case '/':
         return 2;
      case '+':
      case '-':
         return 1;
      default:
         return 0;
   }
}

int applyOp(int a, int b, char op) {
   switch(op) {
      case '+':
         return a + b;
      case '-':
         return a - b;
      case '*':
```

```
            return a * b;
        case '/':
            if (b == 0) {
                printf("Error: Division by zero\n");
                exit(EXIT_FAILURE);
            }
            return a / b;
        case '^':
            return (int)pow(a, b);
        default:
            return 0;
    }
}

int evaluate(char *expr) {
    int i;
    int operandStack[100];
    int topOperand = -1;

    char operatorStack[100];
    int topOperator = -1;

    for (i = 0; expr[i] != '\0'; i++) {
        if (expr[i] == ' ')
            continue;

        if (isdigit(expr[i])) {
            int operand = 0;
            while (isdigit(expr[i])) {
                operand = operand * 10 + (int)(expr[i] - '0');
                i++;
```

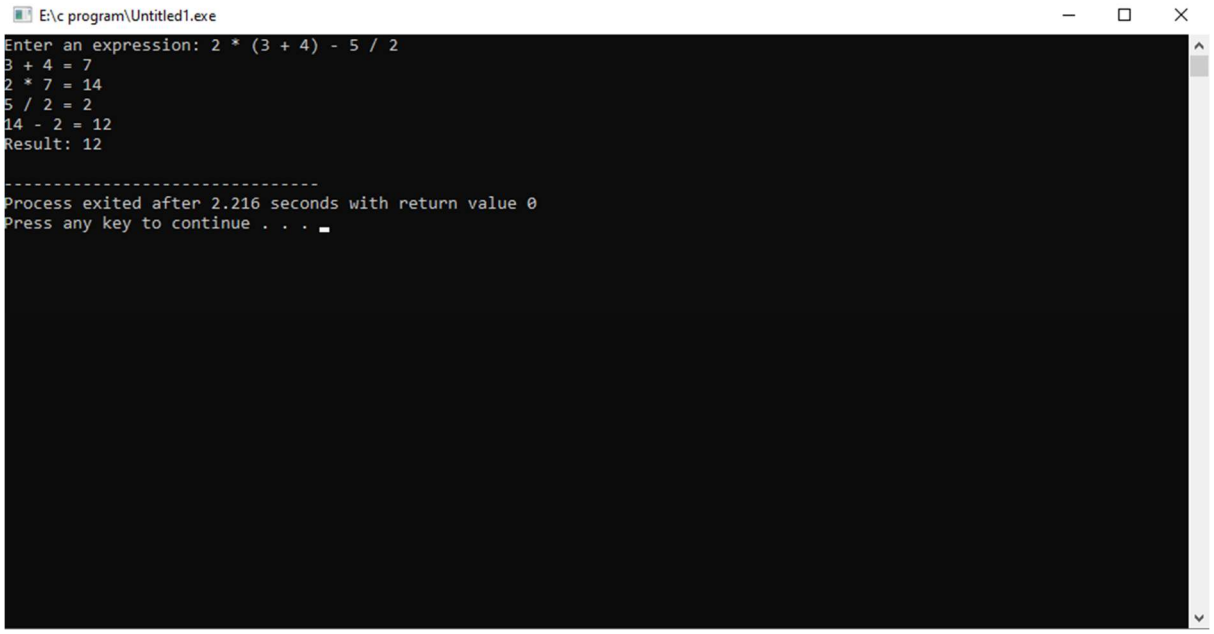
```
            }
            i--;
            operandStack[++topOperand] = operand;
        }
        else if (isOperator(expr[i])) {
            while (topOperator >= 0 && precedence(operatorStack[topOperator]) >= precedence(expr[i]))
{
                int b = operandStack[topOperand--];
                int a = operandStack[topOperand--];
                char op = operatorStack[topOperator--];
                int result = applyOp(a, b, op);
                printf("%d %c %d = %d\n", a, op, b, result);
                operandStack[++topOperand] = result;
            }
            operatorStack[++topOperator] = expr[i];
        }
        else if (expr[i] == '(') {
            operatorStack[++topOperator] = expr[i];
        }
        else if (expr[i] == ')') {
            while (topOperator >= 0 && operatorStack[topOperator] != '(') {
                int b = operandStack[topOperand--];
                int a = operandStack[topOperand--];
                char op = operatorStack[topOperator--];
                int result = applyOp(a, b, op);
                printf("%d %c %d = %d\n", a, op, b, result);
                operandStack[++topOperand] = result;
            }
            topOperator--;
        }
    }
```

```
    while (topOperator >= 0) {
        int b = operandStack[topOperand--];
        int a = operandStack[topOperand--];
        char op = operatorStack[topOperator--];
        int result = applyOp(a, b, op);
        printf("%d %c %d = %d\n", a, op, b, result);
        operandStack[++topOperand] = result;
    }
    return operandStack[topOperand];
}

int main() {
    char expr[100];
    printf("Enter an expression: ");
    fgets(expr, sizeof(expr), stdin);

    int result = evaluate(expr);
    printf("Result: %d\n", result);

    return 0;
}
```


E:\c program\Untitled1.exe
Enter an expression: 2 * (3 + 4) - 5 / 2
3 + 4 = 7
2 * 7 = 14
5 / 2 = 2
14 - 2 = 12
Result: 12

Process exited after 2.216 seconds with return value 0
Press any key to continue . . .