To evaluate our expression we only need to repeatedly handle each input item according to the following cases:

**Numbers.** When a number is encountered in the input, the number is read and pushed onto the numbers stack.

**Operation characters.** When one of the four operation characters is encountered in the input, the character is read and pushed onto the operations stack.

**Right parenthesis.** When a right parenthesis is read from the input, an "evaluation step" takes place. The step takes the top two numbers from the numbers stack and the top operation from the operations stack. These items are removed from their stacks and the two numbers are combined using the operation (with the second number popped as the left operand). The result of the operation is pushed back onto the numbers stack.

**Left parenthesis or blank.** The only other characters that appear in the input are left parentheses and blanks. These are read and thrown away, not affecting the computation. A more complete algorithm would need to process the left parentheses in some way to ensure that each left parenthesis is balanced by a right parenthesis, but for now we are assuming that the input is completely parenthesized in a proper manner.

The processing of input items halts when the end of the input line occurs, indicated by `\n` in the input. At this point, the answer is the single number that remains in the numbers stack.