Evaluation of Postfix

The Postfix notation is used to represent algebraic expressions. The expressions written in postfix form are evaluated faster compared to infix notation as parenthesis is not required in postfix. We have discussed infix to postfix conversion. In this post, the evaluation of postfix expressions is discussed.

Input: str = "2 3 1 * + 9 -"
Output: -4
Explanation:

- Scan 2, it's a number, so push it to stack. Stack contains '2'
- Scan 3, again a number, push it to stack, stack now contains '2 3' (from bottom to top)
- Scan 1, again a number, push it to stack, stack now contains '2 3 1'
- Scan *, it's an operator, pop two operands from stack, apply the * operator on operands, we get 3*1 which results in 3. We push the result 3 to stack. The stack now becomes '2 3'.
- Scan +, it's an operator, pop two operands from stack, apply the + operator on operands, we get 3 + 2 which results in 5. We push the result 5 to stack. The stack now becomes '5'.
- Scan 9, it's a number, so we push it to the stack. The stack now becomes '5 9'.
- Scan -, it's an operator, pop two operands from stack, apply the - operator on operands, we get 5 - 9 which results in -4. We push the result -4 to the stack. The stack now becomes '-4'.
- There are no more elements to scan, we return the top element from the stack (which is the only element left in a stack).

Input: str = "100 200 + 2 / 5 * 7 +"
Output: 757

Algorithm:

```
1. Create an Empty stack st.
2. Traverse through every symbol x of given postfix
    a. If x is and Operand ,push to st.
    b. Else (x is Operator)
        i. Op2=st.pop();
        ii. Op1=st.pop();
        iii. Compute op1 x op2 and push the result
             to st.
3. Return st.top();
```