**3)** (10 pts) ALG (Stacks) Consider evaluating a postfix expression that only contained ***positive*** integer operands and the addition and subtraction operators. (Thus, there are no issues with order of operations!) Write a function that evaluates such an expression. To make this question easier, assume that your function takes an array of integers, `expr`, storing the expression and the length of that array, `len`. In the array of integers, all positive integers are operands while -1 represents an addition sign and -2 represents a subtraction sign. Assume that you have a stack at your disposal with the following function signatures. Furthermore, assume that the input expression is a valid postfix expression, so you don't have to ever check if you are attempting to pop an empty stack. Complete the evaluate function below.

```
void init(stack* s); // Initializes the stack pointed to by s.
void push(stack* s, int item); // Pushes item onto the stack pointed
                               // to by s.
int pop(stack* s); // Pops and returns the top value from the stack
                   // pointed to by s.

int eval(int* expr, int len) {

    stack s;
    init(&s);
    int i;

    for (i=0; i<len; i++) {              // 1 pt
        if (expr[i] > 0)                 // 1 pt
            push(&s, expr[i]);           // 1 pt
        else {
            int op2 = pop(&s);           // 1 pt
            int op1 = pop(&s);           // 1 pt
            if (expr[i] == -1)           // 1 pt
                push(&s, op1+op2);       // 1 pt
            else
                push(&s, op1-op2);       // 2 pts (1 pt for order)
        }
    }

    return pop(&s);                      // 1 pt
}
```