Faculty of Computer Science, IBA

Data Structures (3+1)

Objective:

- To gain understanding of the implementation of STACK data structure with all necessary operations.

Tasks

Stack Application: (Build your own stack)

Postfix notation also known as reverse polish notation in which operator comes after its operands and parenthesis free representation of arithmetic expression. Moreover, expression solve from left to right and no need of priority checks. Due to these advantages, infix arithmetic expressions are first converted to postfix notation and then solve the expression to compute the result. Follow the code structure for Question 1 and 2.

Note: sample code structure in the attached file.

Question 1.

Implement a program that take arithmetic expression as input in infix notation (operator between two operands) then convert it to postfix notation (operator after two operands) using stack (use your array stack implementation). See text book for details.

Input: ===== [(A + B) / B + A - C + (B - C)]	Output: ====== A B + B / A + C - B C - +

Question 2.

Implement a program to compute the result of postfix arithmetic expression using stack.

Input: (Where A=3, B=2, C=1) ===== A B + B / A + C - B C - +	Output: ===== 3

Postfix evaluation (sample code structure for Question 2)

```
stk = the emty stack; while (not end
of input) {
        symb = next input character of exp; if (symb is an
        operand) Push(symb);
        else {
                Opnd2=pop(stk);
                Opnd1=pop(stk);
                Value = result by applying symb to opnd1 and opnd2; Push(value);
          } }
    return (pop(stk));
```