Assignment No. 7

Stack ADT as a Linked List

Aim
Write a program to implement stack as an abstract data type using linked list and use this ADT for conversion of infix expression to postfix, prefix and evaluation of postfix/prefix expression.

Objective(s)	
1	To study basics of stack
2	To learn the stack operations
3	To understand the concept of stack ADT as a linked list

Theory

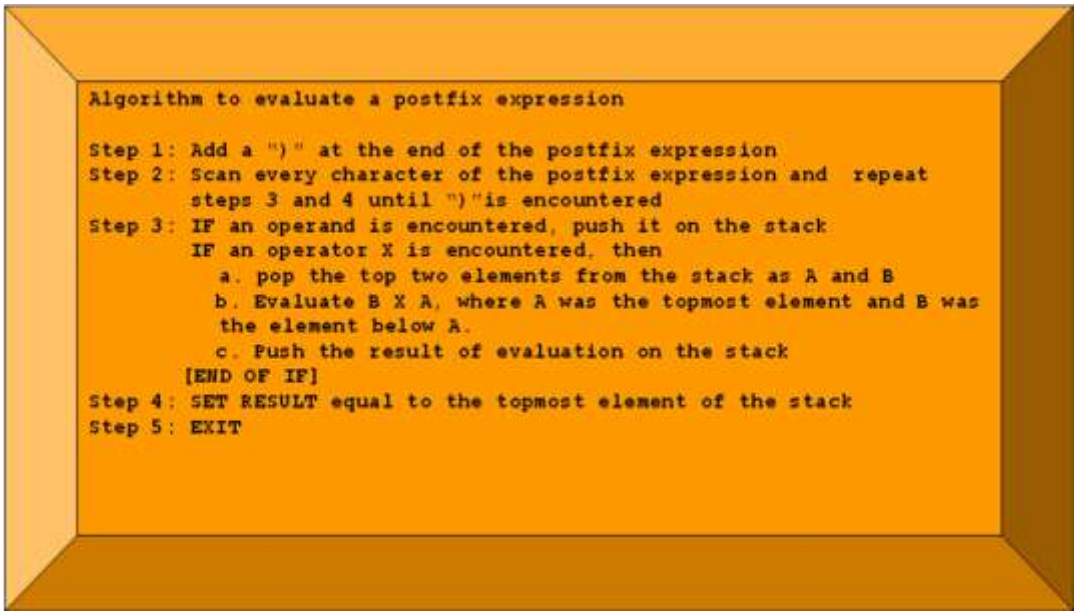
1. What is stack overflow and underflow?
2. Differentiate between: Array and Stack.
3. How a stack implemented using a linked list differs from a stack implemented using an array?
4. How stacks are used in a non-recursive program?
5. Explain: Infix, Prefix and Postfix Expressions.
6. Convert the following infix expressions to their equivalent postfix expressions:
   a) A + B * C / (E - F)
   b) (A ^ B * (C + (D * E) - F)) / G
   c) (A + (B * C - (D / E ^ F) * G) * H)

Algorithms:

a) Infix to Postfix Conversion:

```
Algorithm to convert an Infix notation into postfix notation
Step 1: Add ")" to the end of the infix expression
Step 2: Push "(" on to the stack
Step 3: Repeat until each character in the infix notation is scanned
          IF a "(" is encountered, push it on the stack
          IF an operand (whether a digit or an alphabet) is encountered,
          add it to the postfix expression.
          IF a ")" is encountered, then;
          a. Repeatedly pop from stack and add it to the postfix expression
          until a "(" is encountered.
          b. Discard the "(". That is, remove the "(" from stack and do not
          add it to the postfix expression
          IF an operator X is encountered, then;
          a Repeatedly pop from stack and add each operator (popped from
          the stack) to the postfix expression which has the same
          precedence or a higher precedence than X
          b. Push the operator X to the stack
Step 4: Repeatedly pop from the stack and add it to the postfix expression
        until the stack is empty
Step 5: EXIT
```

b) Evaluation of Postfix Expression:



```
Algorithm to evaluate a postfix expression

Step 1: Add a ")" at the end of the postfix expression
Step 2: Scan every character of the postfix expression and  repeat
        steps 3 and 4 until ")"is encountered
Step 3: IF an operand is encountered, push it on the stack
        IF an operator X is encountered, then
          a. pop the top two elements from the stack as A and B
          b. Evaluate B X A, where A was the topmost element and B was
          the element below A.
          c. Push the result of evaluation on the stack
        [END OF IF]
Step 4: SET RESULT equal to the topmost element of the stack
Step 5: EXIT
```

Conclusion