Introduction to Computing
Stack Use: Postfix Convert and Evaluate
Ramesh Yerraballi

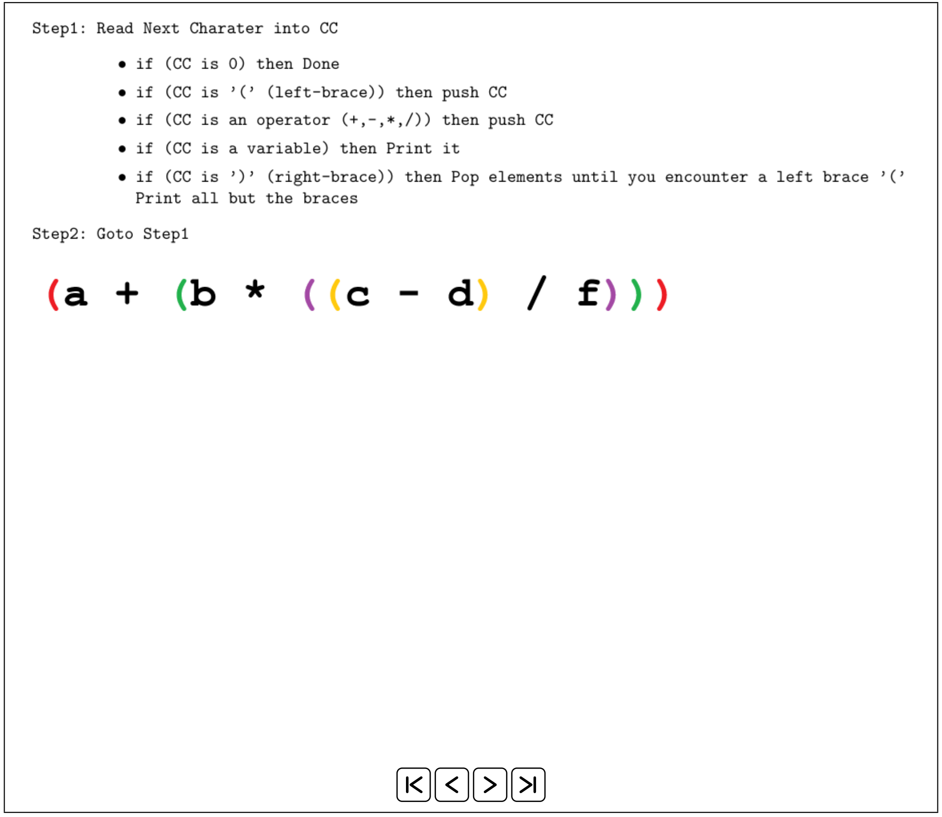
Arithmetic expressions are a fundamental feature in high level programming languages. The common form in which we write these is the human-friendly *infix* notation, where the operator is in between the operands. For example, we write the addition of two variables `x` and `y` as, `x + y`. More complex expressions can be written similarly - `a+b*(c-d)/f`. We can write the same expression by fully parenthesizing it as: `(a + (b * ( (c-d)/f )))`. Unlike humans, computers and more specifically software use the *postfix* notation for both representation and evaluation of arithmetic expressions. Let's look at both these use cases, where a *Stack* is used for processing. Recall that *Push* and *Pop* are the two operations one performs on a *Stack*.

Infix to Postfix

Problem Statement: Given a fully-parenthesized *infix* expression, convert it to its equivalent *postfix* form.
Algorithm: *Assume that the InFix expression is given as a null-terminated string and all variables are single characters. The converted postfix will be stored in the string PostFix*

```
1. Read next character CC from InFix
     • if (CC is NULL) then goto Step 3
     • if (CC is '(' (left-brace)) then push CC on Stack
     • if (CC is an operator (+,-,*,/)) then push CC on Stack
     • if (CC is a variable) then write it to Postfix
     • if (CC is ')' (right-brace)) then Pop from Stack write to Postfix until a left brace (Note:
       do not write braces)
2. Goto Step 1
3. Write NULL to Postfix | Done
```

Here is an illustration of the algorithm on the infix expression: `(a + (b * ( (c-d)/f )))`



```
Step1: Read Next Charater into CC
     • if (CC is 0) then Done
     • if (CC is '(' (left-brace)) then push CC
     • if (CC is an operator (+,-,*,/)) then push CC
     • if (CC is a variable) then Print it
     • if (CC is ')' (right-brace)) then Pop elements until you encounter a left brace '('
       Print all but the braces
Step2: Goto Step1
```

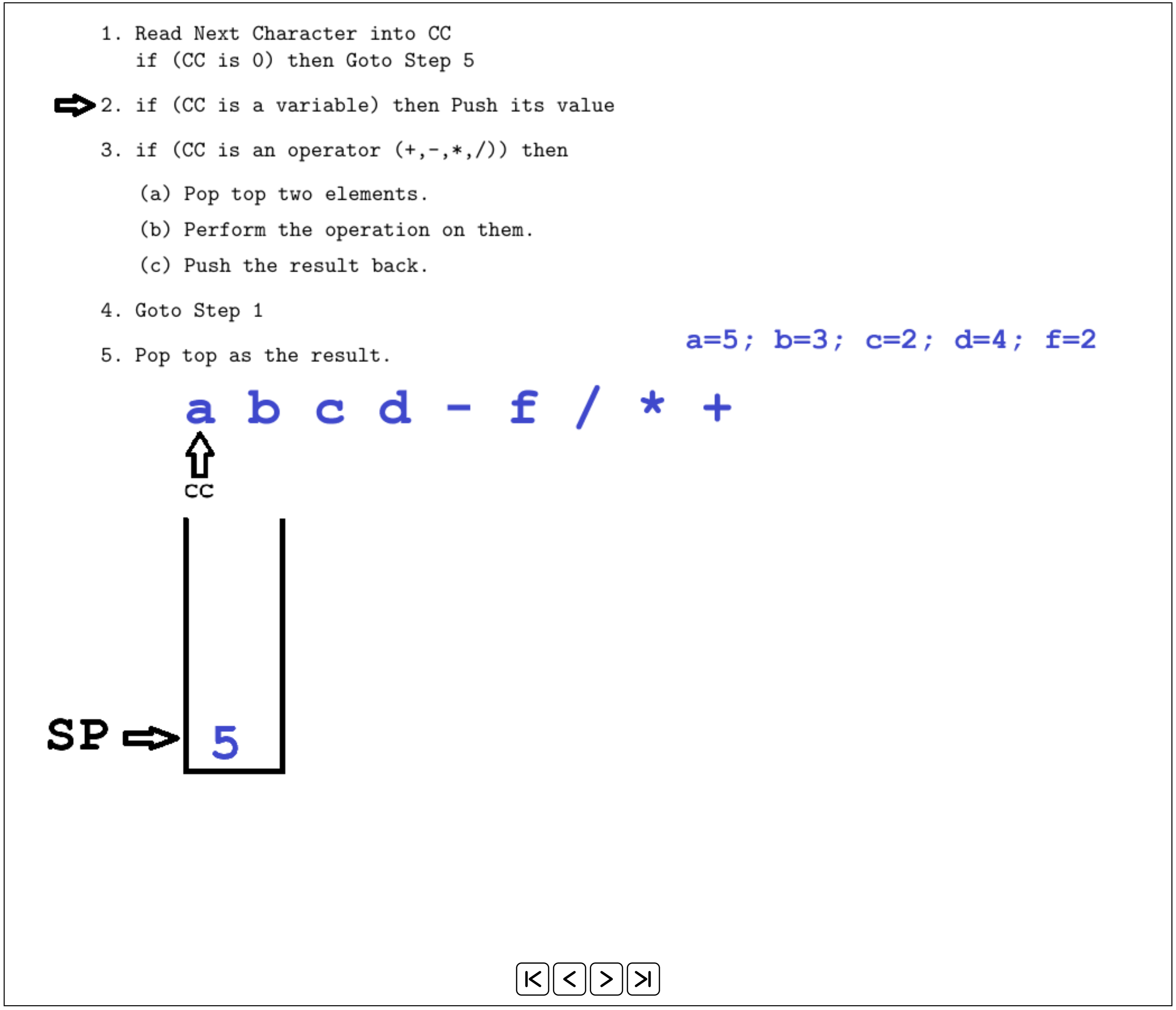
(a + (b * ((c - d) / f)))

Evaluate a Postfix expression

Problem Statement: Given an arithmetic expression in the *postfix* form, evaluate its value for specific values of the variables. **Algorithm**: *Assume that the postfix expression is given as a null-terminated string: PostFix; The result of the evaluation will be stored in a variable name Result*

```
1. Read Next Character from PostFix into CC
   if (CC is NULL) then Goto Step 5

2. if (CC is a variable) then Push its value on the Stack

3. if (CC is an operator X (+,-,*,/)) then

    (a) Pop top two elements off the Stack.
    (b) Perform operation X.
    (c) Push result on the Stack.

4. Goto Step 1

5. Pop value from Stack and write to Result
```

Here is an illustration of the algorithm on the postfix expression: `abcd-f/*+`
for the assigned variable values of: `a=5; b=3; c=2; d=4; f=2`



```
  1. Read Next Character into CC
     if (CC is 0) then Goto Step 5

⇨ 2. if (CC is a variable) then Push its value

  3. if (CC is an operator (+,-,*,/)) then

      (a) Pop top two elements.
      (b) Perform the operation on them.
      (c) Push the result back.

  4. Goto Step 1

  5. Pop top as the result.
```