Evaluating a postfix expression

6 2 3 + - 3 8 2 / + *

Symbol	Opnd1	Opnd2	Value	Opstk
6				6
2				6,2
3				6,2,3
+	2	3	5	6,5
-	6	5	1	1
3				1,3
8				1,3,8
2				1,3,8,2
/	8	2	4	1,3,4

Symbol	Opnd1	Opnd2	Value	Opstk
+	3	4	7	1,7
*	1	7	7	7

Evaluation of Expressions

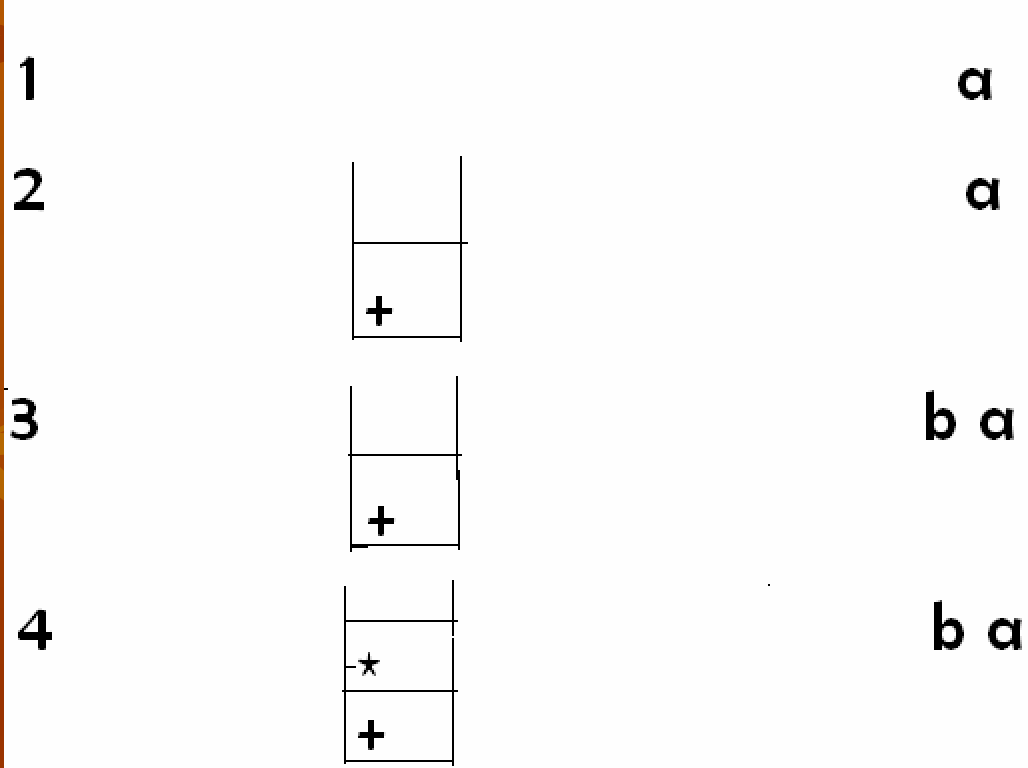
```
opstk = the empty stack;

/* scan the input string reading one element at a time to symbol*/

while ( not end of input){
  symbol=next input character;
  if (symbol is an operand) push (opndstk, symbol);
  else{
        // symbol is an operator
        opnd2 = pop (opndstk);
        opnd1 = pop (opndstk);
        value=result of applying symbol to opnd1 & opnd2;
        push (opndstk, value);
      }//end else
}//end while

return (pop (opndstk));
```

Converting an expression from infix to postfix: a + b * c – d /e $



1		a
2	+	a
3	+	b a
4	⋆ +	b a

5	* +	c b a
6	– +	* c b a
7	– +	d * c b a
8	/ – +	d * c b a

9

/

-

+

e d * c b a

11

+-/ed*cba

a b c * d e / - +

Converting Infix to Postfix (Without Parenthesis)

```
opstk=the empty stack;
while (not end of input){
    symb = next input character;
    if (symb is an operand)
      add symb to postfix string;
    else{
          while(!empty(opstk)&& prcd(stacktop(opstk), symb)){
             topsymb = pop (opstk);
             add topsymb to the postfix string;
           }// end while
           push (opstk, symbol);
        }//end else
  }//end while
// ouput any remaining operators
```

```
while (!empty (opstk)){
    topsymb=pop (opstk);
    add topsymb to the postfix string;
}/*end while */
```

Handling of parenthesis:

Prcd (‘(’,op) = FALSE

Prcd (op, ‘(’) = FALSE

Prcd (op, ‘)’) = TRUE

Prcd (‘(’,op) = undefined

(A+B)*C

Symbol	Postfixstring	Opstk
(		(
A	A	(
+	A	(+
B	A B	(+
)	A B +	
*	A B +	*
C	A B + C	*
	A B + C *	