Algebraic Expressions

Stack Use with Fully Parenthesized Expressions

infix prefix postfix expressions

a+b

+ab

ab+

Checking for Balanced Parentheses

Parens are balanced if

1. Each right paren matches an already encountered left paren
2. When the end of the string is reached each left paren is matched

Draft 1 (what's wrong?)

```
bool isBalanced(exp)
{
  for each ch in exp
     if (ch == '(')
      push(ch)
     else
       if (ch == ')')
         pop()
   if (stack is empty)
    return true
   return false
}
```

Draft 2

```
bool isBalanced(exp)
{
  ch = get(character from exp)
  while (bal and not end of exp)
  {
    if (ch == '(')
      st.push(ch)
    else
      if (ch == ')')
        if (!st.isEmpty())
          st.pop()
        else bal = false
    ch = get(character from exp)
  }
 if (bal and st.isEmpty()) then return true
 return false
}
```

Evaluate Postfix Expressions

```
int evalPostFix(exp)
{
  for each ch in exp
    if (ch is an operand)
       push(ch)
    else
      op2 = pop()
      op1 = pop()
      result = op1 ch op2
      push(result)

 return pop()
}
```

Convert infix to postfix

for (each ch in infix):

1. If ch is operand:
   postfix = postfix + ch
2. if ch == ‘(‘ :
   stk.push(ch)
3. if ch is an operator:
   stk.push(ch)
4. if ch == ‘)’ :
   while(stk.peek() != ‘(‘):
   postfix = postfix + stk.peek()
   stk.pop()
   stk.pop()