Chapter 3
Evaluation of Expressions

top: top of the stack for evaluation

x: a token which represents either an operator or an operand

```
Eval(expression e)
{
        x = NEXT_TOKEN(e);
        while(NOT_END_OF_EXPRESSION(x))
        {
                if(IS_AN_OPERAND(x))
                        Push(&top, x);
                else   /* x is an operator */
                {
                        1. Pop the correct numbers of operands for x from the stack;
                        2. Perform the operation (or generate the code);
                        3. Push the result onto the stack;
                }

                x = NEXT_TOKEN(e);
        }
}
```

Infix to Postfix

top: top of the stack for evaluation

x, *y*: tokens which represent either operators or operands

```
Postfix(expression e)
{
    x = NEXT_TOKEN(e);
    while(NOT_END_OF_EXPRESSION(x))
    {
        if(IS_AN_OPERAND(x))
            Output(x);
        else   /* x is an operator */
            switch(x)
            {
                case '(':    Push(&top, '(');
                             break;
                case ')':    do{
                                 y = Pop(&top);
                                 Output(y);
                             } while(y != '(');
                             break;
                default:     while(PRECEDENCE(top) >= PRECEDENCE(x))
                             {
                                 y = Pop(&top);
                                 Output(y);
                             }
                             Push(&top, x);
            }

        x = NEXT_TOKEN(e);
    }

    while(!STACK_IS_EMPTY(top))
    {
        y = Pop(&top);
        Output(y);
    }
}
```