1: Infix to Postfix and Evaluation

Part 1:

Translate from infix to postfix

Initialize a Queue

Initialize a Stack

Read in one expression in infix format; e.g. (a+b)*c

In the following algorithm, assume Next is what the program reads in.

Algorithm: Infix to Postfix

```
while(Next != EOL) {
    switch(content(Next)) {
            case: ALPHABET:  output content(Next) to Queue; break;
            case:  “ ( “:  PUSH content(Next) into Stack; break;
            case:  “ ) “:  POP the content in Stack to Queue until “(“ is
                           reached, POP it but not to Queue; break;
            case:  “operator”:  while(PRE(TOP) >= PRE(Next)){
                                         POP Stack to Queue;
                                    }   // while
                                PUSH content(Next); break;
            Case: “others”: error;
    } // switch
} //while

POP all in Stack to Queue;
```

Then Queue has the postfix expression.
Note: Queue is First In and First Out.

Infix to Postfix: Evaluating postfix

Part 2: To evaluate postfix expression:

```
1. Define a Stack in float.
2. Read from Queue:
   if (content(Next) is ALPHABET):
         PUSH it into Stack;
   if (content(Next) is OPERATOR)
         POP (from Stack) to var V2;
         POP (from Stack) to var V1;
         perform V1 OPERATOR V2, and result to V;
         PUSH V to Stack;
3. When done, Stack has the evaluation.
```