CSc 127B — Introduction to Computer Science II
(McCann)

Expression Tree Creation Algorithm

Notes:

- You don't need to memorize this for the final, but you should understand how it works.
- Reaching the end of the input is considered to be the lowest-precedence operator by this algorithm
- This algorithm employs two stacks, one for operators and one for references to expression subtrees (which are really just operands that have yet to be evaluated).
- This algorithm doesn't know how to handle parentheses or unary operators. It's not difficult to add those features, but this algorithm includes enough to build binary trees, which is the most important part.

```
initialize next_symbol to any legal operator or operand

while next_symbol is not end-of-the-input

    read the next_symbol

    if next_symbol is an operand

        create an operand node
        place next_symbol in the node
        push a reference to the node on the operand stack

    else if the operator stack is empty,
            or top(operator stack) has lower precedence than next_symbol

        push next_symbol onto the operator stack

    else

        while the operator stack is not empty AND top(operator stack) has
        precedence higher than or equal to next_symbol

            pop the top operator from the operator stack
            create a new operator node
            place the popped operator into the node

            pop the top reference from the operand stack
            store that reference into the node's right child reference field

            pop the top reference from the operand stack
            store that reference into the node's left child reference field

            push a reference to the node on the operand stack

        end while

        push next_symbol onto the operator stack

    end if

end while
```