CSI 33 LECTURE NOTES (Ojakian)

Topic 6: Stacks

OUTLINE
(References: 5.1, 5.2)

1. Intro Example

(a) Application to parenthesis problem

PROBLEM 1. *Program parentheses problem with one sort of parentheses (without stacks).*

PROBLEM 2. *How would you program the parentheses problem when you have multiple types of parentheses??*
Consider Stacks ...

2. Basic Operations of Stack

(a) Recall the Magician child class of RPG Character. Note that the spells are a stack.
(b) Two fundamental operations: push and pop
(c) May have a few others: stack size and look-at-top
(d) Program it

PROBLEM 3. *Program the stack.*

PROBLEM 4. *Use a stack to program the parentheses problem when you have multiple types of parentheses.*

***PROBLEM* 5.** *Consider the simple HTML problem from the Homework.*

3. Application to Post-fix Notation

   (a) Infix: Our usual way of writing mathematical expressions.
   (b) Postfix- Read left to right
   (c) When you reach a binary operation:
      i. Perform it on the 2 most immediate values to the left, then
      ii. Replace the operation and two values by the new value
   (d) About Postfix:
      i. The operations are always evaluated left to right (unlike infix)
      ii. No parethesis needed (unlike infix)
      iii. Typically, easier for a compuer to work with postfix

         PROBLEM 6. *See* **Postfix Tutorial at webpage.** *Evaluate it by just reading it left to right.*
   (e) Postfix evaluation using a Stack

      PROBLEM 7. *Do the last problem again, but now following the use of a Stack in* **Postfix Tutorial at webpage.**

      PROBLEM 8. *Evaluate the Postfix expression:* 6 13 + 3 5 − /

      PROBLEM 9. *Program a postfix evaluator using stacks.*
   (f) Translating Infix to Postfix - A simple way.
      i. Fully Bracket the infix expression so that the parentheses alone determine the order of operations
      ii. From inner most to outermost, move each operator to the right of its 2 operands.

         PROBLEM 10. *Convert some Infix to Postfix.*

4. Application to Context Free Grammar

   (a) What it is?
      i. Collection of rules with single left side non-terminal and sequence of right side
      ii. Start with a non-terminal, replacing in some fashion till only terminals are left.
   (b)

      PROBLEM 11. *Create some more rules and run it.*

      PROBLEM 12. *Understand the code, in particular the the use of a Stack.*