CMPSCI 187 Discussion #4: Infix and Postfix Expressions
Individual Handout

David Mix Barrington
3 October 2012

Last Monday in lecture we introduced infix and postfix expressions, and this Friday in lecture we will present the postfix evaluator from Section 3.8 of DJW. Today we will play with translating and evaluating postfix expressions.

Question 1: The first set of questions asks you to evaluate some postfix expressions, where the operator `/` represents Java integer division without remainder. Some are valid and some are not – identify which ones are invalid and why, and evaluate the valid ones:

- (a) `3 2 4 * 7 - 2 9 * - +`
- (b) `7 1 / 2 + 3 / 8 4 - *`
- (c) `6 4 2 7 3 - + * / - 4 2 *`
- (d) `5 1 + 6 3 - / 4 * 7 - 4 2 *`

Question 2: Translate the *valid* postfix expressions from Question #1 into infix expressions.

Question 3: Next we will translate some infix expressions to postfix. Use the Java hierarchy of operations. (If you have time, evaluate these as well.) Remember the rule – find the operator to be executed last, consider the infix expression as "R op S", translate R and S to R' and S' by the same rule, then write "R' S' op".

- (a) `3 * (2 + 4) + 6 * 3 - 4 * 5`
- (b) `2 / 3 * 4 - 7 + 8 * 2 * 6`
- (c) `(1 + 1) * (1 + 1) + (1 + 1) * (1 + 1)`
- (d) `(((1 + 2) * 3 + 4) * 5 + 6) * 7`

Question 4: Every valid postfix expression has one more number than it has operators – explain why.

Question 5: An expression with k operators and $k+1$ numbers (and no parentheses) may or may not be a valid postfix expression. Describe an algorithm (in English) to tell whether it is.