Project 1 / (8 Points)

CMPE 250, Data Structures and Algorithms, Fall 2010

Instructor: A. T. Cemgil
TA’s: Ali Haydar Özür, Umut Şimşekli, Cihat İmamoğlu

Due: 1 Dec Wed, 14:59 sharp

1. Problem Definition

An algebraic expression is a legal combination of operands and operators. There are three different mathematical notations to write an algebraic expression: infix notation, prefix notation (Polish notation), postfix notation (reverse Polish notation). An example showing the same expression using these three different notations are shown below:

Infix : \((3 \ast 5) + 4\)
Prefix : \(+ \ast 354\)
Postfix : \(35 \ast 4+\)

Although infix notation seems most natural, prefix and postfix notations have advantages of not requiring parantheses to resolve ambiguities and having operator precedence inherently.

Assignment

- Write an ANSI C++ program that given an algebraic expression in one format and values for each variable, converts the expression into other two and evaluates the expression.

Input Format

- First word of the first line is one of the following three: in, pre, post.
- Rest of the first line defines a valid expression in the given notation.
- The second line has one positive integer \(N\): the number of variables in the given expression.
- Next \(N\) lines define values for each variable. No two lines will be the same.

Output Format

- The first line has one integer: the result of expression evaluated with the values for variables given in the input.
- The second line is the infix version of the given expression.
- The third line is the prefix version of the given expression.
- The fourth line is the postfix version of the given expression.
Example Input

in A*R-C
3
C 3
A 1
R 2

Example Output

-1
((A*R)-C)
- * A R C
A R * C -

Important Points

- It is very important that you follow exactly the given input/output format.
- You may assume that at any point in your evaluations, you will deal only with 64-bit signed long integers(long long int), i.e you do not have to worry about floating point numbers and overflows. The input data will assure this. Divisions will be integer divisions, e.g., 3/2 = 1 and not 1.5.
- Your program should work with non-parenthesized infix expressions. For this, you must use the conventional operator precedence and associative rules. For example A+B*C/D/E is equivalent to (A+(B*C)/(D/E))).
- There can be at most 26 variables, i.e. the number of letters in English alphabet. All variables will be uppercase.
- The input will not contain any invalid expression.
- The output Infix expressions should not contain any whitespace and must be properly parenthesized.
- Prefix and postfix expressions should have a single space between each consecutive operand and/or operator.
- There will be five operators in the expressions. They are addition(+), subtraction(−), multiplication(∗), division(⁄) and exponentiation(ˆ).
- Input/output will be via standard I/O, i.e not from any file. This can be achieved via redirection on the command line Prompt> myproj2 < indata.txt > outdata.txt.

Bonuses

- Modify your program so that it works with arbitrary length integers.
- Draw the expression tree in postscript.

Helpful Links

- http://en.wikipedia.org/wiki/Polish_notation
Submission & Grading

- **Submission**
  - You should submit your project in electronic form.
  - You should prepare three input files for demonstration and run your program using these input files. Then, you should get a screenshot for each output. You can get the screenshot using "Print Screen" key on the keyboard and you can paste the screenshot to Paint software which is included in Windows.
  - You should compress your source code (.cpp and .h files) and screenshots (.jpg or .bmp) in a zip file, name it as [pr-][#][student ID] (e.g. pr_1_2000700803.zip), and email to buempe250@gmail.com.
  - Your zip file should NOT contain any executable file (.exe) or any folder. If you sent multiple e-mails, only the last one will be taken into account.
  - Projects that do not contain any screenshot will not be run even if they work correctly.
  - The deadline is Dec 1, Wednesday, 14:59, sharp. Emails tagged later wont be considered.

- **Grading**
  - Your program will be graded based on the quality of your source code and correctness of your output. Correctness of your output will be tested automatically so make sure you stick with the format described above.
  - There are several issues that makes a code piece ‘quality’. In our case, you are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively. Minimize code duplication.
  - Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.
  - Try to write as efficient (both in terms of space and time) as possible. Informally speaking, try to make sure that your program completes under 2 seconds.
  - We may assign the next project before the deadline of this so start as early as possible.