Homework #3

TAs' email: dsata AT csie DOT ntu DOT edu DOT tw

RELEASE DATE: 03/22/2013
DUE DATE: 04/09/2013 (**Tuesday**!!!), 3:30pm (after class)

As directed below, you need to submit your code to the designated place on the course website.

Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

Both English and Traditional Chinese are allowed for writing any part of your homework (if the compiler recognizes Traditional Chinese, of course). We do not accept any other languages. As for coding, either C or C++ or a mixture of them is allowed.

This homework set comes with 200 points and 30 bonus points. In general, every homework set of ours would come with a full credit of 200 points.

3.1 Asymptotic Complexity

In this problem, you can use any theorems in the textbook and any theorems on the class slides as the foundation of your proof. You **cannot** use any other theorems unless you prove them first.

(1) (10%) Do Exercise R-4.24 of the textbook.

(2) (10%) Do Exercise R-4.26 of the textbook.

(3) (10%) Do Exercise C-4.8 of the textbook. (Note: need "proof"!)

(4) (10%) Do Exercise C-4.22 of the textbook.

(5) (10%) Prove or disapprove the following statement. "For non-negative functions f, g, h, if $f(n) = O(g(n))$, then $f(n) + h(n) = O(g(n) + h(n))$."

(6) (10%) Prove or disapprove the following statement. "If $a < 1$ or ($a = 1$ and $b \leq 0$), then $2^{an^2+bn+c} = O(2^{n^2})$."

(7) (10%) Do Exercise R-4.7 of the textbook.

(8) (10%) Do Exercise C-4.16(b) of the textbook.

3.2 Calculators

In this problem, you will be asked to implement two calculators: an "integer calculator" that works on 4-byte integers and supports the arithmetic and bitwise operations in C; a "matrix calculator" that works on 4-byte integer matrices and supports some of the operations in MATLAB.

(1) (60%) Implement the (signed) integer calculator (`hw3_2_1.{c, cpp}`). You can also assume that the input will contain characters only from numbers, the needed operators, and ignore all other characters (including space). You need to implement the following operations with the correct precedence and associativity:

- multiplicative `*`, division `/` or modulo `%`
- binary add `+` or subtract `-`
- bitwise and `&`, exclusive or `^`, or `|`, left shift `<<` or right shift `>>`
- parentheses `()`
- unary minus `-` or plus `+`
- bitwise not ∼

Your program should satisfy the following requirements.

- keep allowing the user to input a line of (i.e. read a line of length $\leq 1000,000$ from `stdin`) infix expression that supports all the operations above on integers, until no more lines can be read (EOF)
- show your stack operations on how to transform the infix expression to a postfix one
- show the corresponding postfix expression
- show the evaluated result, which should be exactly the same as the result computed by a usual C statement on the same expression

You can assume that the expressions provided will be valid, non-ambiguous and non-conflicting to the usual C expression. For instance, something like `i++-j` will not be used to test the program.

Please prepare THREE test cases that help the TA verify all the requirements above. Then, print out the output of your program on those three cases in the written part.

(2) (Bonus 30%) Extend your integer calculator above to handle basic assignment `=` and the assignment versions (when proper) of all the operators you have implemented (`hw3_2_2.{c, cpp}`). You can assume that 26 variables of names 'a', 'b', $\cdots$, 'z' have been declared and initialized to 0 when the integer calculator starts running. No other variables can be used for the calculator.

Please prepare THREE test cases that help the TA verify all the requirements above. Then, print out the output of your program on those three cases in the written part.

(3) (60%) MATLAB is a commercial software for doing matrix calculations. A matrix in MATLAB is declared with the following syntax `[1 2 3; 4 5 6; 7 8 9; 10 11 12]` which indicates a 4 by 3 matrix. You can assume that the matrix contains only integers. You can also assume that the input will contain characters only from numbers, the needed operators and syntax symbols, and you can ignore all other characters. There will be no operations inside the matrix declaration (except for the unary `+` or `-`). You need to implement the following operations with the correct precedence. Note that MATLAB operations are always left-associative. You can check the precedence here `http://www.mathworks.com/help/matlab/matlab_prog/operators.html#f0-38155`

- parentheses `()`, for indicating the highest precedence
- transpose `'`, where `[1 2 0; 3 4 1]'` results in `[1 3; 2 4; 0 1]`. Note that MATLAB actually use conjugate transpose, but given that we are working with integers, there is no difference between the conjugate transpose and the usual transpose.
- component-wise power `.^`, that takes a matrix and a integer. For instance, `[1 2; 3 4] .^ 2` results in `[1 4; 9 16]`.
- unary plus `+`, where `+[1 2; 3 4]` results in `[1 2; 3 4]`
- unary minus `-`, where `-[1 2; 3 4]` results in `[-1 -2; -3 -4]`
- component-wise multiplication `.*`, which operates on two same-size matrices. For instance, `[1 2; 3 4] .* [5 6; 7 8]` results in `[5 12; 21 32]`
- matrix multiplication, `*`, which operates on two matrices such that the number of columns of the first matrix equals the number of columns of the second matrix. For instance, `[1 2; 3 4; 0 1] * [5 6; 7 8]` results in `[19 22; 43 50; 7 8]`. The same operator is used for multiplying an integer to a matrix or a matrix to an integer. For instance, `[1 2; 3 4] * 3` results in `[3 6; 9 12]`.

- matrix addition/subtraction, `+` or `-`, which operates on two matrices of the same size. For instance, where `[1 2; 3 4] + [5 6; 7 8]` results in `[6 8; 10 12]`. The same operator is used for adding/subtracting an integer to/from a matrix or a matrix to/from an integer. For instance, `[1 2; 3 4] - 3` results in `[-2 -1; 0 1]`.
- colon operator `:`, which creates a matrix of one row that contains an arithmetic progression. For instance, `1:3:11` results in `[1 4 7 10]`. For simplicity, you can assume that the middle operand will always be positive.

Your program should satisfy the following requirements.

- keep allowing the user to input a line of (i.e. read a line of length $\leq 1000,000$ from `stdin`) infix MATLAB expression that supports all the operations above on integers and integer matrices, until no more lines can be read (EOF)
- show the evaluated result, which should be exactly the same as the result computed by a usual MATLAB statement on the same expression

You can assume that the expressions provided will be valid and non-ambiguous.

Please prepare THREE test cases that help the TA verify all the requirements above. Then, print out the output of your program on those three cases in the written part.

For all problems, you need to "show the evaluated result" by printing out a single line of the form for the integer calculator

```
RESULT: 1126
```

or for the matrix calculator

```
RESULT: [5 5 6 6; 1 2 3 4]
```

You can freely decide how you want to show your other parts (stacks, postfix) by printing out other lines.

Submission File (Program) and Written Copy

Please upload your program as a single ZIP compressed file to CEIBA before the deadline at 3:30pm on Tuesday (04/09/2013). The zip file should be like `b86506054.zip`, where the file name should be changed to your own school ID. The ZIP file should contain the following items:

- all the source code for your program.
- a Makefile to compile your code and run your program. The TAs will type `make` to compile your source files to two (or three) programs. Then they will `make run1` to test your homework 2.2(1), `make run2` to test your homework 2.2(2) (if possible), `make run3` to test your homework 2.2(3). In your Makefile, you should let each of your program redirect from a file '`./input`' as its `stdin` and redirect to a file '`./output`' as its `stdout`.
- an optional `README`, anything you want the TAs to read before grading your code

For all the problems that require illustrations, please submit a written (or printed) copy in class or to CSIE R217 before the deadline.

MEDAL USAGE: If you want to use the gold medals for this homework, please write down the number of medals that you want on the first page of your printed copy (something like "use 2 medals"). Use your medals wisely—usage cannot be retracted.