ECE220: Computer Systems & Programming **Fall 2018 ZJUI**
Lab 13 **Tuesday 11 December / Thursday 13 December**

Reading Lines and Evaluating Expressions

In today's lab, you must write a simple recursive function to evaluate an expression consisting of a tree of nodes. Nodes can be either binary arithmetic operators or numbers. The expressions are built for you from strings read from **`stdin`** (you must also write the line-reading code). The goal is for you to see a simple version of the recursive tree-walking code that you must write for the MP11 compiler. The problem here is quite simple, so you may want to also take a look at how the trees are built if you have extra time.

Begin by checking out the **`lab13`** subdirectory in your Subversion repository. The directory contains a copy of this document (**`lab13.pdf`**), a C header file **`lab13.h`**, a C source file **`lab13main.c`** that provides most of the code, a **`Makefile`**, and a C source file, **`lab13.c`**, with which you can start this lab. You can immediately "**`make`**" the executable, but without your code, the program does nothing.

What the Given Code is Doing

The code given to you builds an expression as a tree of nodes. Given a string, the recursive function identifies whether the string starts with a number or a parenthetical expression, then builds a tree by creating number and operator nodes as necessary. You have also been given a **`main`** program (in **`lab13.c`** this time) that calls all of the support routines, once you add the line of code needed to read a line of input from **`stdin`**.

The Task

First, complete the **`main`** function by fixing the condition on the **`while`** loop. The loop should continue so long as a new line of input is available from **`stdin`**. The line should be read into the array buf. The loop processes the resulting string by building an expression from it, printing the expression, using the code you will write next to evaluate the expression, and then freeing the expression. Once **`stdin`** ends, the loop should terminate.

Now for the main task: implement the function **`evaluate_expression`**, in **`lab13.c`**. The function is invoked for each expression built from the input and must return the correct value for that expression, which is then printed to the display.

The signature for your function is as follows:

```
double evaluate_expression (node_t* expr);
```

The expression is a tree of nodes. The node structure and all necessary definitions appear in the header file **`lab13.h`**. Use recursion to evaluate the two operands of operator nodes. The value of **`NUMBER`** nodes can be used directly.

As mentioned earlier, this lab is quite easy, so you may want to spend the rest of the time looking at the **`build_expression`** and **`do_build_expression`** functions and the supporting subroutines in **`lab13main.c`**. This code builds the expression trees that your code evaluates. You may want to walk through the code in a debugger to see how it works after typing in an interesting expression, such as "**`3 + 8 * 7 - 17`**".