The University of Texas at San Antonio
Computer Science Division
San Antonio, Texas 78249

CS 1723, Data Structures
Fall Semester, 1998
Programming Assignment 2, Due September 11, 1998
Reverse Polish Evaluation

The Assignment: For this assignment you are to write a program, `evaluate`, that will find the value of a reverse Polish expression.

The input source (RPN) will be made up of the following elements:

- single-digit integer constants: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, or `9`.
- operators: $\wedge$, `* /`, `+`, or `-`.
- special symbol to terminate the input: #.

This assignment limits to single-digit integer constants to keep things simpler.

For example, here is sample input source, which calculates one root of the equation $y = x^2 + 3x + 2$, with $a = 1, b = 3, c = 2$:

```
32^41*2*-12/^3-21*/#
```

Overall organization: Your program should be contained in a single directory, say `assign2`. Within this directory there will be several files implementing the the main function and the stack.

Details about the program `evaluate.c`: Organize this program as (at least) *three* files: `evaluate.c`, `estack.h`, and `estack.c`.

The evaluation of one of these RPN strings can proceed as described in the text and in class: Use an evaluation stack. Operands are pushed on the stack, and operators pop their arguments off the stack and push the result. When the final # is encountered, the remainder of the stack is popped and printed. (It should just be a single value.)

The arithmetic should be done using `doubles`. Thus the output of the second example should be the following:

```
-1.000000
```

A single character that is a digit can be converted to a `double` with

```
#include <ctype.h>
...
if (isdigit(c))
        return (double) (c - '0');
...
```

The raise-to-a-power operator can be handled with the built-in function `pow(x, y)` (see the white book, page 251). For this to work you need to include `<math.h>`, and (on runner) you need to add `-lm` to the compiler options, so that the math library will be searched, as shown in the `makefile` below. The `lint` program also needs this option as shown—without it `lint` will produce 600 lines of warnings.

The new `makefile` for `evaluate.c`:

```
# makefile for evaluate program
evaluate:  evaluate.c estack.c estack.h
        cc -g -o evaluate evaluate.c estack.c -lm
lint:
        lint -m -u evaluate.c estack.c -lm
```

Required Execution:

Your program *must* execute the following test data, with the expected output:

Input test data	Expected output
23+4*#	20.000000
234+*#	14.000000
52^34*2^+12/^#	13.000000
55+2^13/^#	4.641589
32^41*2*-12/^3-21*/#	-1.000000