Program Three, Just some Cool computations.

Cool Programming

Write a moderately complex cool program.

This assignment is worth 25 points.

Goals

When you finish this homework, you should:

- Have begun to learn the cool programing language.

Formal Description

(Mostly taken from the COOL assignment package)

This assignment asks you to write a short Cool program. The purpose is to acquaint you with the Cool language.

A machine with only a single stack for storage is a {\em stack machine}. Consider the following very primitive language for programming a stack machine:

Command	**Meaning**
int	push the integer *int* on the stack
+	push a `+' on the stack
s	push an `s' on the stack
e	evaluate the top of the stack (see below)
d	display contents of the stack
x	stop

The `d' command simply prints out the contents of the stack, one element per line, beginning with the top of the stack. The behavior of the `e' command depends on the contents of the stack when `e' is issued:

- If `+' is on the top of the stack, then the `+' is popped off the stack, the following two integers are popped and added, and the result is pushed back on the stack.
- If `s' is on top of the stack, then the `s' is popped and the following two items are swapped on the stack.
- If an integer is on top of the stack or the stack is empty, the stack is left unchanged.

The following examples show the effect of the `e' command in various situations; the top of the stack is on the left:

stack before	**stack after**
+ 1 2 5 s ...	3 5 s ...
s 1 + + 99 ...	+ 1 + 99 ...
s 1 + 3 ...	1 + 3 ...

You are to implement an interpreter for this language in Cool. Input to the program is a series of commands, one command per line. Your interpreter should prompt for commands with `<`. Your program need not do any error checking: you may assume that all commands are valid and that the appropriate number and type of arguments are on the stack for evaluation. You may also assume that the input integers are unsigned. Your interpreter should exit gracefully; do not call `abort()` after receiving an `x`.

You are free to implement this program in any style you choose. However, in preparation for building a Cool compiler, we recommend that you try to develop an object-oriented solution. One approach is to define a class `StackCommand` with a number of generic operations, and then to define subclasses of `StackCommand`, one for each kind of command in the language. These subclasses define operations specific to each command, such as how to evaluate that command, display that command, etc. If you wish, you may use the classes defined in `atoi.cl` in the `~dbennett/cool/examps` directory to perform string to integer conversion. If you find any other code in this directory that you think would be useful, you are free to use it as well.

Discussion

Submission

Please submit a source code file containing all of your source code.