CS 565 Out: August 28, 1996
Jens Palsberg Due: September 4, 1996

Homework 2:

The Stack Machine

1. (10 pts) Read Section 2.3 of *EOPL* and do exercise 2.3.10 on page 62.

2. (10 pts) Stack machines are ideally suited for executing expressions in so-called "reverse Polish" (or postfix) notation, in which one writes

$$e_1\ e_2\ \texttt{+}$$

instead of the infix notation

$$e_1 + e_2$$

or the prefix notation

$$\texttt{(}\ \texttt{+}\ e_1\ e_2\ \texttt{)}$$

Write a program which takes a list of the following form:

$$\begin{aligned}\langle\text{exp}\rangle ::= &\ \langle\text{number}\rangle \\ ::= &\ \texttt{(}\ \texttt{+}\ \langle\text{exp}\rangle\ \langle\text{exp}\rangle\ \texttt{)} \\ ::= &\ \texttt{(}\ \texttt{-}\ \langle\text{exp}\rangle\ \langle\text{exp}\rangle\ \texttt{)} \\ ::= &\ \texttt{(}\ \texttt{*}\ \langle\text{exp}\rangle\ \langle\text{exp}\rangle\ \texttt{)} \\ ::= &\ \texttt{(}\ \texttt{/}\ \langle\text{exp}\rangle\ \langle\text{exp}\rangle\ \texttt{)}\end{aligned}$$

and translates it into postfix code suitable for execution on the stack machine (see lecture note 2 and the file `stack-machine.scm` on the CS 565 homepage). Test your translator by executing its output on the stack machine. Your program might translate `(+ (* 3 4) 5)` to something like

```
(push 3 push 4 mult push 5 add halt)
```

Modify the code in `stack-machine.scm` to add new opcodes to the stack machine to handle multiplication, subtraction, and division, and test your translator by running its output on the extended stack machine.

3. (Extra credit, 5 pts) Extend your stack machine to include loops. This might be done by using a construct like `while`, or by including some provision for labels and gotos in your stack machine code. This is a design exercise; it is intended to test your ingenuity.

Demonstrate your design by writing and running a stack machine program that reads two numbers (assumed non-negative integers) from the input and multiplies them by doing repeated addition.