CMPE 230 Systems Programming
Homework 1 (due March 24th)
(This project can be done using Java or C or C++)

In this project, you will implement

a) A compiler for a language called Simple that will compile Simple code to an abstract *stack machine* code.
b) A stack machine simulator that will take stack machine code and execute it.

The grammar for the Simple language will be as follows:

stm → **id :=** *expr*
| **print** *expr*
| **if** *expr* **then** *stm*
| **while** *expr* **do** *stm*
| **begin** *opt_stmts* **end**

opt_stms → *stmt_list*
| **ε**

stmt_list → *stmt_list* **;** *stm*
| *stm*

expr → *term moreterms*

moreterms → **+** *term moreterms*
| **-** *term moreterms*
| **ε**

term → *factor morefactors*

morefactors → ***** *factor morefactors*
| **/** *factor morefactors*
| **div** *factor morefactors*
| **mod** *factor morefactors*
| **ε**

factor → **(** *expr* **)**
| **id**
| **num**

Your compiler should be able to parse codes given in Simple language following the grammar rules given above. Note that **id** is an identifier (variable) and **num** is a number.

The stack machine will be able to execute the following instructions:

+ - * / div mod	binary operations
push v	push v onto the stack

rvalue l	Push contents of data location l
lvalue l	Push address of data location l
pop	Throw away value on top of the stack
:=	The r-value on top is placed in the l-value below it and both are popped
copy	Push a copy of the top value on the stack
print	Print the value on top and then pop the value.
label l	Target of jumps to l, has no other effect
goto l	Next instruction is taken from statement with label l
gofalse l	Pop the value on top, jump to l if it is zero
gotrue l	Pop the value on top, jump to l if it is nonzero
halt	Stop execution

Here are some example of translations of small code fragments:

Example 1

The program **val := (461*y) div 4 + (200*m+2) div 5 + d** is translated to:

Instructions

0	lvalue val
1	push 461
2	rvalue y
3	*
4	push 4
5	div
6	push 200
7	rvalue m
8	*
9	push 2
10	+
11	push 5
12	div
13	+
14	rvalue d
15	+
16	:=
17	halt

Data

18 (val)	**Val**
19 (y)	
20 (m)	
21 (d)	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
...	

Stack

...	
...	
...	
...	
...	
4093	
4094	
4095	

As seen in this example, infix expressions are converted into postfix expressions.

Example 2

Consider the following if statement

if *expr* then *stm*

It will be translated as:

code for *expr*
gofalse out
code for *stm*
label out

Example 3

Consider a while loop:

while expression do stm

It will be translated as follows:

label test
code for *expr*
gofalse out
code for *stm*
goto test
label out