Write the answers to this homework set into a file named hw5, and submit it using the submit script, by the date and time shown above.

1. (3) Question 21 of the recent Midterm exam contains a Python program. Run that program. Include both your version of the program and the run (as you did for echo.py) in your answers to this homework set.

2. (6) Write a LOSL program to calculate the \( n^{th} \) Fibonacci number for a given \( n \). Recall,

\[
\begin{align*}
Fibonacci(n) &= 1 \text{ if } n \leq 2 \\
Fibonacci(n) &= Fibonacci(n-1) + Fibonacci(n-2) \text{ if } n > 2
\end{align*}
\]

For example, \( Fibonacci(8) = 21 \).

You may use the Common Lisp program /projects/shapiro/CSE305/losl as an LOSL interpreter while you develop your program, but you should turn in just your program.

More specifically, your program must include the variables \( N \) and \( F \), and may use additional variables. The variable \( N \) should be initialized before your program runs, and all other variables must be initialized by your program. When your program stops, the variable \( F \) must contain \( Fibonacci(N) \).

The LOSL interpreter uses the same syntax as the LOSL defined in the course web pages and on page 8 of the recent Midterm exam, except that the symbol “;” is replaced by “>”.

The following trace shows the LOSL interpreter running a program to increment whatever value is initially stored in $x$ by 1. The value $x$ is initialized, before the program runs, to 3. The trace shows the program being run without tracing, followed by a dump of memory, then being run with tracing. This trace starts after running Common Lisp. How to do that will be explained further in recitation.

```
cl-user(1): :ld /projects/shapiro/CSE305/losl ; Fast loading /projects/shapiro/CSE305/losl.fas1
cl-user(2): :pa losl

losl(3): (run '(x x fetch 1 + store pop stop x> 3))
DONE

losl(4): (dump)
-------------
Symbol Table
-------------
x: 8
PC: 8
Stack: nil
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 4
-------------

losl(5): (run '(x x fetch 1 + store pop stop x> 3) :trace t)
-------------
Symbol Table
-------------
x: 8
PC: 0
Stack: nil
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3
-------------
```
Symbol Table
------------
x: 8
PC: 1
Stack: (8)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3

Symbol Table
------------
x: 8
PC: 2
Stack: (8 8)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3

=======
Symbol Table
-------------
x: 8
PC: 3
Stack: (3 8)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3

Symbol Table
-------------
x: 8
PC: 4
Stack: (1 3 8)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3

Symbol Table
------------
x: 8
PC: 5
Stack: (4 8)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 3

Symbol Table
------------
x: 8
PC: 6
Stack: (4)
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 4
=========
Symbol Table
--------
x: 8
PC: 7
Stack: nil
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 4
=========

=========
Symbol Table
--------
x: 8
PC: 8
Stack: nil
RAM
---
0: x
1: x
2: fetch
3: 1
4: +
5: store
6: pop
7: stop
8: 4
=========
DONE
losl(6):