Kelvin Mei
COSI121B
PS1 Writeup

1.17

Double and halve are functions given by the problem. In this exercise, I'm to write a method (log a b) where it multiplies a and b recursively and returns the output. My base case is if (= b 0), return a 0. Else, if b is even, I recursively call (log (double a) (halve b)). If b is odd, I add a to the recursive call of (log (a) (- b 1)). This uses a logarithmic number of steps because the even case divides b by 2. I also have a check to check if a and b are negative or not, because the logarithmic algorithm only works for b > 0. If a and b are both negative, I turned them positive. If only b is negative, I swapped a and b.

1.34

When (f f) is called, it yields (f 2)
(f 2) then yields (2 2)
This results in an error because it's neither a procedure nor an expression.

1.43

This was actually shown in class

1.44

Repeated was given in class/was in 1.43. Smooth takes a function with argument x and a dx and finds the average of (f(x-dx), f(x), f(x+dx)). Nfold takes an integer n and repeats the smooth function n number of times. The output is a function. Smooth takes in 3 inputs, f, dx, and x, where f is the function.

Function Doubling

In the substitution model, it looks like
((double 1+) 0)
(1+ (1+ 0))
= 2
The next one takes this function and calls the double function on that, which squares the number of times 1+ is performed.

Scheme cannot calculate ((((((double double) double) double) double) 1+) 0) because of the max integer limit. The maximum integer limit is 2147483647 or 2^31 - 1.
In the pattern above, it is 2^1, 2^2, 2^4, 2^16. The next integer is 2^256, which exceeds the limit by far.

Lab

1:
I added a check to check for (> fuel-burn-rate (/ (fuel ship-state) dt). If it's burning too much and there isn't enough fuel, it'll use the remaining as the burn rate instead. Also prevents fuel from going below zero.

2: Added procedure parameter to the methods play and lander-loop. Also changed lander-loop to cond because I like cond better than if.

3: Added the (define (random-choice a b) line to the given code, changed full-burn and no-burn to a and b.

4: Similar to problem 3, except there is an extra parameter h and the conditional is that if the height is greater than h, do strategy 1, else do strategy 2.

5: random-choice and height-choice are similar, they call choice with two strategies and a Boolean as input. Choice then looks at the Boolean and performs the appropriate strategy based on the Boolean being true or false.

6: (play (height-choice no-burn (random-choice full-burn ask-user) 40))
This will allow the ship to drop until it gets below 40 height. Then it will have a 50% chance of asking the user for the burn-rate or just full-burn.

7: If the ship is to land at a perfect speed of 0, then velocity final is 0.
Using the kinematics equation $v_f^2 = v_i^2 + 2ah$, we can prove that the acceleration needed is $v^2/2h$
$v_f^2 = 0$, so $0 = v_i^2 + 2ah$. Solve for a and you get $-v_i^2/2h = a$. The negative is because it's accelerating in the opposite direction.

8: Given the formula, I just applied it to the scheme function. However, I had to divide it by dt to get the velocity and add gravity to the final result to compensate for the gravity pulling down. This one does not work when number 10 changes update, because it limits the fuel-burn-rate.

9: Letting the ship drop to 20 saves more fuel than dropping to 30 and then constant-acc.

```
(height 1.6653345369377348e-016 velocity -1.1102230246251565e-016 fuel -96.49999999999997)
(height 1.1102230246251565e-016 velocity -5.551115123125783e-017 fuel -96.74999999999997)
(height 8.326672684688674e-017 velocity -5.551115123125783e-017 fuel -96.99999999999997)
(height 5.551115123125783e-017 velocity -5.551115123125783e-017 fuel -97.24999999999997)
(height 2.7755575615628914e-017 velocity -5.551115123125783e-017 fuel -97.49999999999997)
(height 0.0 velocity 0.0 fuel -97.74999999999997)
0.0
good landing
game-over
(height 1.1102230246251565e-016 velocity -5.551115123125783e-017 fuel -43.74999999999999)
(height 8.326672684688674e-017 velocity -5.551115123125783e-017 fuel -43.99999999999999)
(height 5.551115123125783e-017 velocity -5.551115123125783e-017 fuel -44.24999999999999)
(height 2.7755575615628914e-017 velocity -5.551115123125783e-017 fuel -44.49999999999999)
(height 0.0 velocity 0.0 fuel -44.74999999999999)
0.0
good landing
game-over
>
```

Without a fuel limit, a perfect landing would require about 53 more fuel if you start at 30 instead of 20.

10: I created a limit function that takes in a burn-rate and returns 1 if the burn-rate is greater than one. Then I went into my update function and replaced all burn-rate instances with (limit (burn-rate)) so all the burn-rates are limited to 1 and lower.

11: This one is similar to problem 8; I had the burn-rate calculated and then checked if it was greater than one. If it is, then it calls constant-acc. If not, it will free fall with no-burn as the burn-rate until the possible burn-rate for constant-acc is > 1.