PPL Assignment 1.1

1.
    a. What is the difference between values and types? How are they related?

    The difference between values and types is that a type defines what values a variable can have. A value is what a variable can be.

    b.
        i. (define x 5) - causes a side effect that variable x is added to the GE
        ii. (+ x 1) - displays 6 as a side effect but doesnt change the value of x
        iii. (if (= x 6)

            (display ‘mary-had-a-little-lamb)

            (* x x)

    displays 25 (* x x) since the previous snippet didnt change the value of x

2.
    a. (> 3 2 1) - [number*number*number -> boolean]
    b. ((lambda (x y) (- (* x x) (* y y)))) - [number*number -> void], ananymous function
    c. ((lambda (x y) (- (* x x) (* y y))) 4 -7) - [number*number-> number]
    d. (+) - [void -> number]
    e. (define x 42) - procedure, binds x to 42 in GE
    f. (number? 5) - [scheme type -> boolean]

3.
    a. (+ (/ 1 0) 1 (/ 1 2)) - error! Division by 0 not allowed
    b. (* 2) - 2
    c. (+ x 42) - error! Unidentified variable x
    d. (define x (/ 1 3)) - adds x to the GE with value 1/3
    e. (lambda () (4 5)) - procedure [void -> void]
    f. ((lambda (x) (display x) (* 2 x)) 5) displays: 510
    g. (+ x 42)- $42_{1/3}$ x was defined in step d
    h. (/ x) - 3

4.
    a. (/ 1 0) - division by 0 not allowed

b. (= x 5) - x is undefined

5. A

a. (cond ((odd? 72) #f)

```
(#f  (display ‘whoops))
(else (or 7
        (< (/ 1 0) 2)
        #f
        #t)))
```

72 isnt odd so we evaluate the next conditional

#f isnt different from false so we evaluate the next conditional

In else, **7** is returned since it doesnt evaluate to false, the rest of the expressions in the or expression arent evaluated

b. (and #t

```
(or (even? 3)
        (lambda () (+ 1 2 3)))
(* 1 2 3))
```

All three expressions in the **and** conditional evaluate to not #f therefore it returns the last evaluated expression which is (* 1 2 3) and evaluates to **6**