```
Addition
(+ 10 1) ; Expected output: 11

Subtraction
(- 2 3) ; Expected output: -1

Multiplication
(* 0 1) ; Expected output: 0
(* 2 3) ; Expected output: 6

Division
(/ 0 10) ; Expected output: 0
(/ 10 5) ; Expected output: 2
(/ 5 2) ; Expected output: 2

Nil?
(nil? ()) ; Expected output: True
(nil? (and 5 ())) ; Expected output: True
(nil? (or () 5)) ; Expected output: False

Number?
(number? 2) ; Expected output: True
(number? (+ 9 (* 2 45))) ; Expected output: True
(number? (1 2 3)) ; Expected output: False

Symbol?
(symbol? append) ; Expected output: True
(symbol? (append newlist 6)) ; Expected output: False

List
(list? newlist) ; Expected output: False (even though it evaluates to
a list, it is a symbol)
(list? (1)) ; Expected output: True
(list? ()) ; Expected output: True (not an atom)

Define and Set
(define increment (a) (+ a 1)) ; function "increment" defined
(set num 2) ; num = 2
(increment num) ; Expected output: 3
(+ num (+ 3 6)) ; Expected output: 11
(set num (increment num)) ; num = 3
(+ num (+ 3 6)) ; Expected output: 12

Arithmetic Expression
(- 314 (* 10 (+ 15 (/ (* 5 9) 3)))) ; Expected output: 14

Define
(define myVar 42) ; Expected output: 42

Set
```

```
(set myVar 24) ; Expected output: 24

Number?
(number? 42) ; Expected output: True

Symbol?
(symbol? mySymbol) ; Expected output: True

Cons
(cons 1 (2 3)) ; Expected output: (1 2 3)

If
(if (= 2 2) true false) ; Expected output: true

Car
(car (1 2 3)) ; Expected output: 1

Cdr
(cdr (1 2 3)) ; Expected output: (2 3)

And
(and (= 2 2) (> 3 1)) ; Expected output: True

Or
(or (= 2 3) (> 3 1)) ; Expected output: True

Not
(not (= 2 3)) ; Expected output: True

List
(list 1 2 3) ; Expected output: (1 2 3)     ;ERROR

Nil?
(nil? ()) ; Expected output: True
```