

This review is very similar in structure to the actual quiz. You will get exactly this reference at the top.

## Reference

```
;; filter (alpha -> boolean) list-of-alpha -> list-of-alpha
;; constructs a list from all items in the list for which the predicate is true
```

```
;; map (alpha -> beta) list-of-alpha -> list-of-beta
;; constructs a list by applying the function to each item in the list
```

(**big-bang** *state-expr clause ...*)

Where clause is one or many of:

- (**on-tick** *tick-expr*) where *tick-expr* : (WorldState -> WorldState)
- (**on-key** *key-expr*) where *key-expr* : (WorldState key-event -> WorldState)
- (**to-draw** *render-expr*) where *render-expr* : (WorldState -> scene)
- (**stop-when** *last-world?*) where *last-world?* : (WorldState -> boolean)

**1. Higher-order functions.** Write the following functions using higher-order functions:

```
;; squares list-of-numbers -> list-of-numbers
(check-expect (squares (list 2 3 4)) (list 4 9 16))
```

```
;; first-quadrant list-of-posn -> list-of-posn
;; returns a list of positions in the first quadrant (positive x and y coordinates)
(check-expect (first-quadrant (list (make-posn 1 3)
                                   (make-posn -1 3)))
              (list (make-posn 1 3)))
```

```
;; intersection list-of-numbers list-of-numbers -> list-of-numbers
;; returns a list of all numbers in both the input lists
```

**2. Designing worlds.** Build a 'flight simulation' in a window 400 pixels high and 800 pixels wide. When the program starts the image of a plane is at the upper left. As time goes by the plane glides down and to the right until it touches the ground and stops.

```
(define PLANE (text ">>" 20 "black"))  
(define WIDTH 800)  
(define HEIGHT 400)
```