```
(require 2htdp/image)
(require 2htdp/universe)

;; =================
;; =================
;; Constants:

(define BOARD-SIZE 10)   ; The board is 10 cells by 10 cells

(define CELL-PIXELS 14)                          ; cells are square
(define BOARD-WIDTH (* BOARD-SIZE CELL-PIXELS)) ;
(define BOARD-HEIGHT BOARD-WIDTH)                ;

;; Images for the head and body elements of the snake
;; as well as for the food.
;; (NOTE: in the first version you only use HEAD.)

(define HEAD (circle (/ CELL-PIXELS 2) "solid" "green"))
(define BODY (circle (/ CELL-PIXELS 2) "solid" "red"))
(define FOOD (circle (/ CELL-PIXELS 2) "solid" "blue"))

(define MTS (empty-scene BOARD-WIDTH BOARD-HEIGHT))

;; =================
;; =================
;; Data Definitions:

; *** DO THIS STEP FOR PRE-LAB ***
 Step 1 (but AFTER reading through the whole file):

 The following data definitions have only type comments, or only a
 define-struct and type comment. Complete these data definitions
 with examples, templates and template rules used.

 When you are done, print out a copy of your completed data definitions
 and draw the reference/self-reference arrows. Arrows should start from
 where they are called to the appropriate types comments. Label your
 arrows with either R (reference) or SR (self-reference).


;; Direction is one of:
;;  - "U"
;;  - "D"
;;  - "L"
;;  - "R"
;; interp. the four directions a snake could travel
;; <Examples redundant for an enumeration>

(define (fn-for-dir d)
  (cond [(string=? d "U") (...)]
        [(string=? d "D") (...)]
        [(string=? d "L") (...)]
        [(string=? d "R") (...)]))

;; Template rules used:
;;  - one of: 4 cases
;;  - atomic distinct: "U"
;;  - atomic distinct: "D"
;;  - atomic distinct: "L"
;;  - atomic distinct: "R"
```

```
50
51
52  (define-struct cell (c r))   ; c and r stand for column and row
53  ;; Cell is  (make-cell Integer[-1, BOARD-SIZE] Integer[-1, BOARD-SIZE])
54  ;; interp. a cell position on the board from top-left corner
55  ;;         -1 and BOARD-SIZE are on the edges of the board and indicate
56  ;;         "going out of bounds"/game-over condition
57  (define C1 (make-cell -1 -1))
58  (define C2 (make-cell -1 BOARD-SIZE))
59  (define C3 (make-cell BOARD-SIZE -1))
60  (define C4 (make-cell BOARD-SIZE BOARD-SIZE))
61  (define C5 (make-cell (/ BOARD-SIZE 2) (/ BOARD-SIZE 2)))
62
63  (define (fn-for-cell c)
64    (... (cell-c c) (cell-c r)))
65
66  ;; Template rules used:
67  ;;  - compound: 2 fields
68  ;;  - atomic non-distinct: Integer[-1, BOARD-SIZE]
69  ;;  - atomic non-distinct: Integer[-1, BOARD-SIZE]
70
71
72  (define-struct snake (dir head))
73  ;; Snake is (make-snake Direction Cell)
74  ;; interp. a snake with a head moving in some direction
75  (define S1 (make-snake "U" (make-cell 1 1)))
76  (define S2 (make-snake "L" (make-cell 3 2)))
77  (define S3 (make-snake "R" (make-cell 2 1)))
78
79  (define (fn-for-snake sn)
80    (... (fn-for-dir (snake-dir sn))
81         (fn-for-cell (snake-head sn))))
82
83  ;; Template rules used:
84  ;;  - compound: 2 fields
85  ;;  - reference: (snake-dir sn) is Direction
86  ;;  - reference: (snake-head sn) is Cell
87
88
89  (define-struct game (snake))
90  ;; Game is (make-game Snake) ; later on we will add fields to game
91  ;; interp. the game state with the snake
92  (define G1 (make-game (make-snake "D" (make-cell 0 1))))
93  (define G2 (make-game (make-snake "L" (make-cell 3 2))))
94  (define G3 (make-game (make-snake "R" C5)))
95
96  (define (fn-for-game gm)
97    (... (fn-for-snake (game-snake gm))))
98
99  ;; Template rules used:
100 ;;  - compound: 1 field
101 ;;  - reference: (game-snake gm) is Snake
```