1. Descarga el código del proyecto final que acompaña este archivo y ábrelo en DrRacket.
2. Completa el cuerpo de la función (all-but-last segs) que dada una lista, regresa dicha lista sin el último elemento.
   Por ejemplo: (all-but-last '(1 2 3 4)) =>'(1 2 3)

Solución:

```
;; (cons X [Listof X]) -> [Listof X]
;; Returns a List that is does not contain the last element of the given list.
;; > (all-but-last '(1 2 3 4))
;; '(1 2 3)
(define (all-but-last segs)
  #| Aquí va su código. |#
  (cond
    [(= (length segs) 1) empty]
    [else (cons (first segs) (all-but-last (rest segs)))]))
```

3. Completa el cuerpo del predicado (pos=? p1 p2) que indica si dos posiciones son iguales.
   Por ejemplo: (posn=? (posn 1 1) (posn 1 1)) =>#t

Solución:

```
;; Posn Posn -> Boolean
;; Are the two posns are equal?
;; > (posn=? (posn 1 1) (posn 1 1))
;; true
(define (posn=? p1 p2)
  #| Aquí va su código. |#
  (and  (= (posn-x p1) (posn-x p2))
        (= (posn-y p1) (posn-y p2))))
```

4. Completa el cuerpo de la función (snake-head sn) que obtiene la cabeza de la viborita. Recuerda que una viborita está definida como una lista de posiciones (segmentos).
   Por ejemplo: (snake-head (snake right"(list (posn 1 1) (posn 2 1)))) =>(posn 1 1)

Solución:

```
;; Access the head position of the snake.
;; snake-head : Snake -> Seg
;; > (snake-head (snake "right" (list (posn 1 1) (posn 2 1)))
;; (posn 1 1)
(define (snake-head sn)
  #| Aquí va su código. |#
  (car (snake-segs sn)))
```

5. Completa el cuerpo de la función (snake-body sn) que obtiene el cuerpo de la viborita. Recuerda que una viboritaestá definida como una lista de posiciones (segmentos).

   Por ejemplo: (snake-body (snake right"(list (posn 1 1) (posn 2 1)))) =>(list (posn 2 1))

Solución:

```
;; Snake -> [Listof Segs]
;; returns the snake's body.
;; That is everyting that isn't the snake's head.
(define (snake-body sn)
  #| Aquí va su código. |#
  (cdr (snake-segs sn)))
```

6. Completa el cuerpo de la función (snake-change-dir sn d) que dada una viborita y una dirección, cambia la dirección dela misma.

   Por ejemplo: (snake-change-dir (snake right"(list (posn 1 1))) "left") =>(snake "left"(list (posn 1 1)))

Solución:

```
;; Snake Direction -> Snake
(define (snake-change-dir sn d)
  #| Aquí va su código. |#
  (snake d (snake-segs sn)))
```

7. Completa el cuerpo del predicado (dir? x) que dado un valor, indica si es una dirección valida.

   Por ejemplo: (dir? üp") =>t

Solución:

```racket
;; String -> Boolean
;; Is the given value a direction?
;; > (dir? "up")
;; #t
(define (dir? x)
  #| Aquí va su código |#
 (cond
    [(equal? x "up") #t]
    [(equal? x "down") #t]
    [(equal? x "left") #t]
    [(equal? x "right") #t]
    [else #f]))
```

8. Completa el cuerpo del predicado (oppsite-dir? d1 d2) que dadas dos direcciones, indica si son opuestas.

   Por ejemplo: (opposite-dir? üpdown") =>t

Solución:

```racket
;; Direction Direction -> Boolean
;; Are d1 and d2 opposites?
;; > (opposite-dir? "up" "down")
;; #t
(define (opposite-dir? d1 d2)
  #| Aquí va su código |#
  (cond
    [(and (equal? d1 "up") (equal? d2 "down")) #t]
    [(and (equal? d1 "down") (equal? d2 "up")) #t]
    [(and (equal? d1 "left") (equal? d2 "right")) #t]
    [(and (equal? d1 "right") (equal? d2 "left")) #t]
    [else #f]))
```

9. Una vez completadas las funciones, ejecuta el archivo con el botón de ejecutar que se encuentra en la barra superior de DrRacket Si al ejecutar el archivo se muestra una pantalla como la de la Figura 1, entonces tus funciones pasaron todas las pruebas y puedes jugar. Si lanza un error, corrígelo y vuelve a ejecutar el archivo.