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

#;
 Instructions:

 Click the mouse to the left or right of the zombie to make him turn towards
 the mouse click.

 Use the space bar to reset the zombie back the start position.


;; My zombie program

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

(define SPEED 2)

(define WIDTH 600)
(define HEIGHT 150)

(define CTR-Y (/ HEIGHT 2))

(define MTS (empty-scene WIDTH HEIGHT "white"))

(define ZOMBIE-RIGHT  )
(define ZOMBIE-LEFT  )

(define START (+ (/ (image-width ZOMBIE-RIGHT) 2) 2))


;; =================
;; Data definitions:

(define-struct zs (x o))
;; ZS is (make-zs Integer Integer)
;; A zombie that follows your mouse
;;  x is the x coordinate in pixels
;;  o is the orientation left or right 1 or -1
(define Z1 (make-zs START 1))    ;zombie facing right moving left to right --->
(define Z2 (make-zs 500 -1))     ;zombie facing left moving right to left <---

#;
(define (fn-for-zs zs)
  (... (zs-x zs)   ;Ingeger
       (zs-o zs))) ;Integer

;; Template rules used:
;;  - compound: 2 fields


;; =================
;; Functions:

```

```
;; ZS -> ZS
;; start the world with (main (make-zs START 1))
;;
(define (main zs)
  (big-bang zs                          ; ZS
            (on-tick   next-zombie)     ; ZS -> ZS
            (to-draw   render-zombie)   ; ZS -> Image
            (on-mouse  handle-mouse)    ; ZS Integer Integer MouseEvent -> ZS
            (on-key    handle-key)))    ; ZS KeyEvent -> ZS


;; ZS -> ZS
;; advance to next x-coordinate
(check-expect (next-zombie Z1)
              (make-zs (+ (zs-x Z1) (* (zs-o Z1) SPEED))
                       (zs-o Z1)))
(check-expect (next-zombie Z2)
              (make-zs (+ (zs-x Z2) (* (zs-o Z2) SPEED))
                       (zs-o Z2)))
(check-expect (next-zombie(make-zs 300 -1))
              (make-zs (+ 300 (* -1 SPEED)) -1))

;(define (next-zombie ws) (make-zs 0 1)) ;stub
;same template as ZS

(define (next-zombie zs)
  (make-zs (+ (zs-x zs) (* (zs-o zs) SPEED))
           (zs-o zs)))

;; ZS -> Image
;; render the image in the proper position on the MTS
(check-expect (render-zombie Z1) (place-image ZOMBIE-RIGHT START CTR-Y MTS))
(check-expect (render-zombie Z2) (place-image ZOMBIE-LEFT 500 CTR-Y MTS))

;(define (render-zombie ws) MTS) ;stub
;same template as ZS

(define (render-zombie zs)
  (cond [(equal? (zs-o zs) 1) (place-image ZOMBIE-RIGHT
                                           (zs-x zs)
                                           CTR-Y MTS)]
        [else                 (place-image ZOMBIE-LEFT
                                           (zs-x zs)
                                           CTR-Y
                                           MTS)]))


;; ZS Integer Integer MouseEvent -> ZS
;; change zombie state on mouse entering the scene
;;  zombie will turn to face mouse click direction

;(define (handle-mouse zs x y me) (make-zs START 1)) ;stub
;same template as ZS

(define (handle-mouse zs x y me)
  (cond [(mouse=? me "button-down") (make-zs (zs-x zs)
                                             (orient-zombie(zs-x zs) x))]
        [else zs]))

;; ZS KeyEvent -> ZS
;; reset the zombie back to the beginning
(check-expect (handle-key Z2 " ") (make-zs START 1))
(check-expect (handle-key Z2 "a") Z2)
```

```
112
113 ;(define (handle-key zs ke) (make-zs START 1)) ;stub
114 ;same tempate as ZS
115
116 (define (handle-key zs ke)
117   (cond
118     [(key=? ke " ") (make-zs START 1)]
119     [else zs]))
120
121
122 ;; Integer Integer -> Integer
123 ;; returns 1 if first integer is less than second
124 ;; returns -1 if first integer is greater
125 (check-expect (orient-zombie 0 600) 1)
126
127 ;(define (orient-zombie x m) 1) ;stub
128 ;(define (orient-zombie x m)    ;the template
129 ;  (... x m))
130
131 (define (orient-zombie x m)
132   (if (< x m)
133       1
134       -1))
135
136
137 (main (make-zs START 1))
```