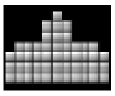
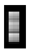
```
(require 2htdp/image)

;;;;;;;;;;;;;;;;;; SI proj ;;;;;;;;;;;;;;;;;;
;;;              Yohei Yasukawa           ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Definitions for testing
(define                        )
(define                            )
(define                            )
(define                            )
(define                            )
(define                            )
(define                            )
(define                            )
(define                            )
(define                            )
(define                                )

; Definitions for Init
(define                 )
(define                  )
;(define EMPTY_SCENE (empty-scene BG_WIDTH BG_HEIGHT))
(define
       )

;;;;;;;;;;;;;;;;
;;; Defender ;;;
;;;;;;;;;;;;;;;;

; Definitions for Defender
```



```
(define               ) ; 50x40 pixels
(define                ) ; how fast a defendder moves to right
and left.
(define                 )
(define                 )
(define                                  )
(define                                               )
(define                          )


;; defender-key: Defender Key -> Defender
; calculates the state following the given state if given
```

```
key is pressed








;; touch-left-wall? : Posn -> boolean
; determine if a given position is touching a wall on the
left







;; touch-right-wall? : Posn -> boolean
; determine if a given position is touching a wall on the
right









;; move-left : Defender -> Defender
; move a given defender to the left in 1 px









;; move-right : Defender -> Defender
```

```
; move a given object to the right in 1 px









; defender-render : Defender -> image
; constructs an image representing the given state

  (place-image DEF_IMG
               current
               DEF_POSY
               BG_BLANK)


;;;;;;;;;;;;;;;;
;;; Barrage  ;;;
;;;;;;;;;;;;;;;;

; Definitions for Barrage
(define            )
(define            )
(define
```

```
)
(define                )
(define         )
(define
```

```
) ; 10x20 pixels
(define









                       )
(define                  )
(define                                 )

;; missile-tick: Missile -> Missile
; calculates the state following the given state if only
time passes
```

```
;; barrage-tick: Barrage -> Barrage
; calculates the state following the given state if only
time passes
```

```
;; barrage-filter-offscreen : Barrage -> Barrage
; filters out a offscreen missile from a given barrage if
it exists
```

```
;; touch-top-wall? : Missile -> boolean
; determine if a given missile is touching a wall on the
top.
```

```
;; move-up : Missile -> Missile
; move a given object to the top in 1 px
; barrage-render : List of Missile -> image
; constructs an image representing the given state

  (cond
    [(empty? barrage) BG_BLANK]
    [(cons? barrage) (cons-barrage-render barrage)]
    )

  (overlay (place-image MISSILE_IMG
                        (posn-x (first barrage))
                        (posn-y (first barrage))
                        BG_BLANK)
           (barrage-render (rest barrage))
           )


;;;;;;;;;;;;;
;;; Alien ;;;
;;;;;;;;;;;;;

; AlienLoc is a posn
; the direction the Alien will move next
; Ex:


; AlienDir is a srting, one of
; - "left"
; - "right"
; - "down"
; Ex:


; regular-alien is a strucure containing
; - location (AlienLoc)
; - direction (AlienDir)
```

```
(define
        )


; diver-alien is a structure containing
; - location (AlienLoc)
; - direction (AlienDir)
; - diving? (boolean)


(define
              )


; shielded-alien is a structure containing
; - location (AlienLoc)
; - direction (AlienDir)
; - health (integer 0-2; amount of shield remaining)


(define
               )


; Aliens is a List (of Alien)
; the Aliens in a particular game state
(define



        )


;; Definitions for Aliens
```
(define) ; 20x20 pixels

(define) ; 20x20 pixels

(define) ; 20x20 pixels
```
(define              ) ; the space between aliens or
between alien and wall in x axis
(define              ) ; the space between aliens or
between alien and wall in y axis
(define              ) ; show how fast the alien moves in
a game.
(define             )
```

```
(define               )
(define                )
(define                )
(define                             )
(define


                                                )
(define


                                            )
(define


                                                        )
(define


                                                )
(define


                                            )
(define


                                                        )
(define


                                            )
(define

                                                    )
(define                             )


;; alien-next-loc : AlienLoc AlienDir -> AlienLoc
; calculate the next alien location by a given location
and direction.
```

```
;; alien-next-dir : AlienLoc AlienDir -> AlienDir
; calculates the next direction of alien by a given
location and direction.
```

```
;; adjacent-left-wall? : AlienLoc -> boolean
; determine if an alien on a given location is adjacent to
the left wall.
```

```
;; adjacent-right-wall? : AlienLoc -> boolean
; determine if an alien on a given location is adjacent to
the right wall.
```

```
;; aliens-tick : Aliens -> Aliens
; calculates the state following the given state if only
time passes

  (cond
    [(empty? aliens) empty]
    [(cons? aliens) (cons-aliens-tick aliens)]
    )

  (cond
    [(regular-alien? (first aliens)) (cons
(move-regular-alien (first aliens))
                                          (aliens-tick
(rest aliens)))]
    [(diver-alien? (first aliens)) (cons (move-diver-alien
(first aliens))
                                        (aliens-tick (rest
aliens)))]
```

```
    [(shielded-alien? (first aliens)) (cons
(move-shielded-alien (first aliens))
                                           (aliens-tick
(rest aliens)))]
    )

;; move-regular-alien : RegularAlien -> RegularAlien
; calculate the next regular alien state by a given
regular alien state
```

```
;; move-diver-alien : DiverAlien -> DiverAlien
; move a given diver alien to the next location
```

```
;; dived-diver-alien : DiverAlien -> DiverAlien
```

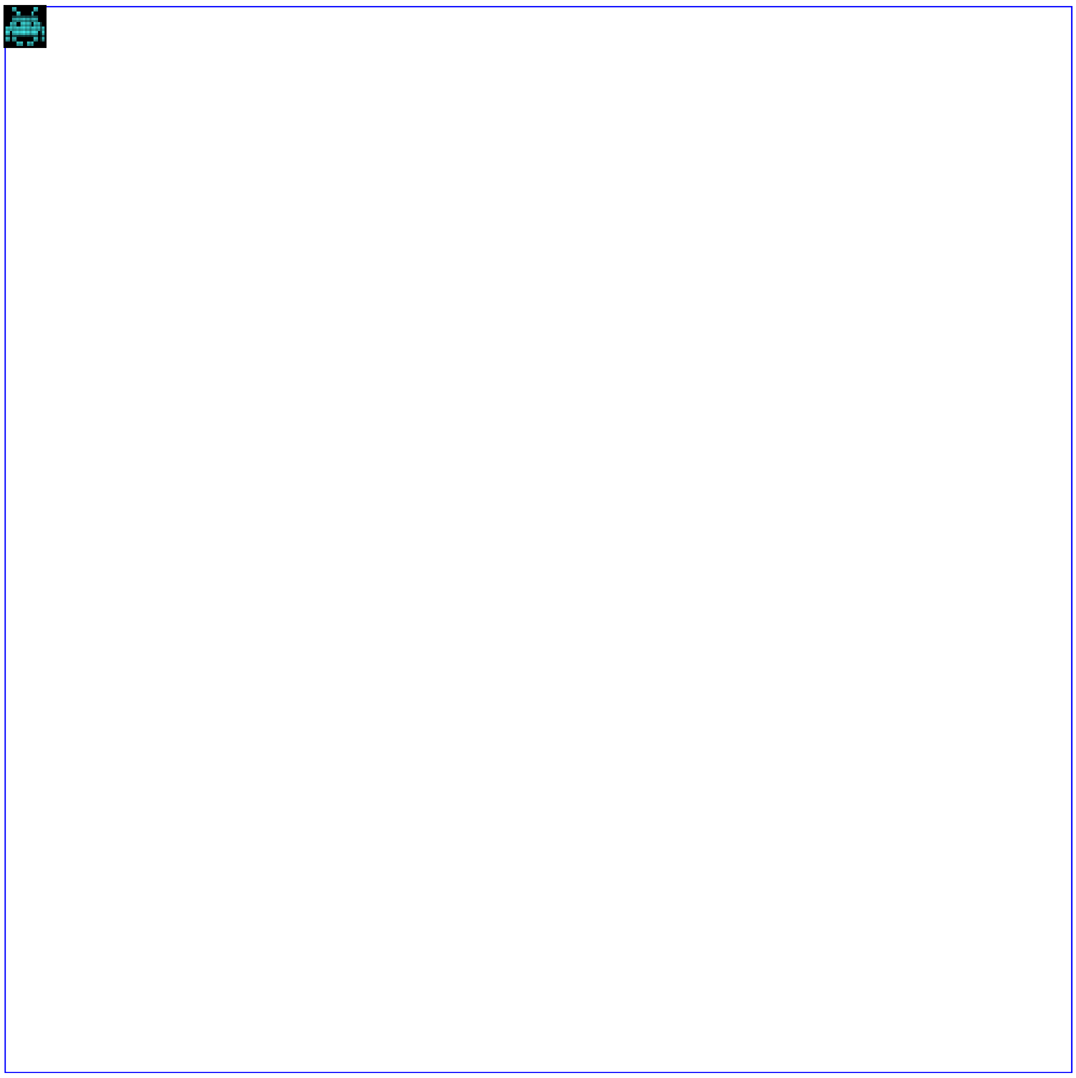
```
; calculate a dived location of a given diver alien.
```

```
;; moved-diver-alien : DiverAlien -> DiverAlien
; calculate a moved location of a given diver alien.
```

```
;; over-line? : Posn Posn -> boolean
; determines if a first given position is over the other
position line.
```

```
;; move-shielded-alien : shieldedAlien -> shieldedAlien
; calculate the next regular alien state by a given
regular alien state
```

```
;; aliens-render : Aliens -> image
; constructs an image representing the given state

  (cond
    [(empty? aliens) BG_BLANK]
    [(cons? aliens) (cons-aliens-render aliens)]
    )


  (cond
    [(regular-alien? (first aliens))
     (overlay (put-image (regular-alien-loc (first aliens))
                         RA_IMG)
              (aliens-render (rest aliens)))]
    [(diver-alien? (first aliens))
     (overlay (put-image (diver-alien-loc (first aliens))
                         DA_IMG)
              (aliens-render (rest aliens)))]
    [(shielded-alien? (first aliens))
     (overlay (put-image (shielded-alien-loc (first
aliens))
                         SA_IMG)
              (aliens-render (rest aliens)))]
    )

;; put-image : location image -> image
; put a given image on a given location
```



```
;;;;;;;;;;;;
;;; Game ;;;
;;;;;;;;;;;;

; Definitions for Game

(define
             )

; game-render : Game -> image
; constructs an image representing the given state

  (overlay (barrage-render (game-barrage game))
           (aliens-render (game-aliens game))
           (defender-render (game-defender game))
           )

;; game-tick : Game -> Game
; calculates the state following the given state if only
time passes

  (make-game
   (game-defender game)
   (aliens-tick (remove-alien-hit-by-missile (game-aliens
game)
                                             (game-barrage
game)))
   (barrage-tick (remove-missile-hit-to-alien
(game-barrage game)
                                              (game-aliens
game)))
   )

;; remove-alien-hit-by-missile : Aliens Barrage -> Aliens
; remove an alien hit by a misile in a given barrage from
a given aliens.
```

```
;; damage-aliens : Aliens Barrage -> Aliens
; damage a first alien in given aliens (decrement health
once)
```

```
;; decrement-health : Alien -> Alien
; decrement health of a first alien of given aliens.
; If the alien has no health or 0 health remained, the
alien died (removed).
```

```
;; remove-missile-hit-to-alien : Barrage Aliens -> Barrage
; remove a missile hit to an alien from a given barrage.
```

```
      (cons (first barrage) (remove-missile-hit-to-alien
(rest barrage) aliens))
```

```
;; missile-hits-aliens? : Missile Aliens -> boolean
; determine if a given missile hits one of given aliens.
```

```
      (missile-hits-aliens? missile (rest aliens))
```

```
;; alien-hit-by-barrage? : Alien Barrage -> boolean
; determine if missiles in a given barrage hit a given
alien.
```

```
;; missile-hits-alien? : Missile Alien -> boolean
; determine if a given missile hits a given alien.
```

```
     (shielded-alien? alien) (missile-hits-around-aloc?
missile

(shielded-alien-loc alien))
```

```
;; missile-hits-around-aloc? : Missile AlienLoc -> Boolean
; determine if a given missile hits an area (20x20 square)
from a given location.
```

```
;; game-key: Game Key -> Game
; calculates the state following the given state if given
key is pressed


                        (make-game
                             (defender-key (game-defender
game) key)
                             (game-aliens game)
                             (new-barrage (game-barrage
game)
                                          (game-defender
game)))
```

```
;; new-barrage: Barrage Defender -> Barrage (List of
Missile)
; create a new missile by given defender status,
; and add it to a given barrage.
```

```
;; new-missile: Defender -> Missile
; create a new missile
; Game functions

          current



;(game-start game-init)



;;;;;;;;;;;;
;;; Main ;;;
;;;;;;;;;;;;

;; Definitions for Main
(define                    )
(define
```

WIN!

"press N to start a new game"

```
                                                  ) ; 500x500
```

```
pixels
(define
```

LOSE
"press N to start a new game"

```
                                                ) ; 500x500
pixels
(define                           )

;; main-tick : Main -> Main (Game)
; calculates the state following the given state if only
time passes

  (cond
    [(string? main) main] ; this condition stops a
mis-evaluating error.
    [(wiped-out-aliens? (game-aliens main)) "win"]
    [(invaded-by-aliens? (game-aliens main)) "lose"]
    [else (game-tick main)]
    )

;; wiped-out-aliens? : Aliens -> boolean
; determine if a given state wiped out aliens.
```

```
;; invaded-by-aliens? : Aliens -> boolean
; determine if any of aliens invaded a dead line.
```

```
;; get-alien-loc : Alien -> AlienLoc
; get a location info from a given alien.
```

```
;; main-key : Main Key -> Main (Game)
; calculates the state following the given state if given
key is pressed

                    (restart-key main key)
```

```
;; restart-key : Main Key -> Main (Game)
; calculates the state following the given state if given
key is pressed

  (cond
    [(string=? key "n") main-init]
    [else main]
    )

;; main-render : Main -> image
; constructs an image representing the given state

  (cond
    [(string? main) (put-result-image main)]
    [else (game-render main)]
    )

;; create-result-image : Main -> image
; put a result image by a given main state.









; Main functions

            current




;(main-start main-init)
```