

Fall 2026 - CS 111 - Exam 1 Reference

NOTE: for all of the exam questions, you are expected to **ASSUME** the following:

- that you are using DrRacket with a Language level of "Beginning Student" or "Beginning Student with List Abbreviations".
- that the following expressions are **ALREADY** in your DrRacket Definitions window and/or .rkt file:

```
(require 2htdp/image)
(require 2htdp/universe)
(require 2htdp/batch-io)
```

- that it **IS okay** to (correctly) use list abbreviation syntax (`list`) in your answers if you would like.
- that the following comments are **already** in your DrRacket Definitions window/.rkt file:

```
; DATA DEFINITION
; a Color is one of:
;   - a string containing the name of a color ("red", "blue", etc.), or
;   - the result of a make-color expression with a red-value, a green-value,
;     and a blue-value, and optionally also a transparency value (each in the interval
;     [0, 255])

; DATA DEFINITION
; a NumOrF is one of:
;   - number
;   - #false

; DATA DEFINITION
; an Anything is an expression of ANY type

; DATA DEFINITION
; a list is one of:
;   - empty
;   - (cons Anything list) ; cons for CONSTRUCT a list

;==== TEMPLATE for a function that needs
;   to "walk through" all of the elements of a
;   variable-length list
;
; (define (a-list-funct ... a-list ...)
;   (cond
;     [(empty? a-list) ...]
;     [else (...
;              ... (first a-list) ...
;              (a-list-funct ... (rest a-list) ...) ...)]
;   )
; )
```

- Now, for a few examples, signatures, and purpose statements, for reference and exam purposes:

```
; signature: get-discount: string -> number
; purpose: expects a customer level ("gold", "silver", or "bronze"), and returns the
;   appropriate discount rate for a customer at that level
```

```

; the following are all #true:

(= 8 (+ 3 5))
(string=? "George" (string-append "Ge" "orge"))
(equal? (circle 30 "outline" "red") (circle (+ 15 15) "outline" "red"))

; signature: string->number: string -> NumOrF
; purpose: expects a string containing digits/numeric characters, and returns the
;         equivalent numeric value in that string. If provided with a value whose characters
;         cannot be easily converted to a number, it returns #false.

; signature: circle: number string Color -> image
; purpose: expects a radius in pixels, either "solid" or "outline", and a
;         color, and returns a circle image with that radius, style, and color

; signature: star: number string Color -> image
; purpose: expects the distance in pixels between points of a desired star image,
;         either "solid" or "outline", and a color, and returns a star image
;         with that size, style, and color

; signature: square: number string Color -> image
; purpose: expects a side-length in pixels, either "solid" or "outline", and a
;         color, and returns a square image with sides of that length, in that style,
;         of that color

; signature: rectangle: number number string Color -> image
; purpose: expects a width and height in pixels, "solid" or "outline", and a
;         color, and returns a rectangle image with that width, height, style, and
;         color

; signature: text: string number Color -> image
; purpose: expects some text, a desired font-size, and a color, and returns
;         an image of that text in that font-size and color

; signature: empty-scene: number number -> scene
; purpose: expects a width and a height in pixels, and returns an empty scene
;         with those dimensions

; signature: place-image: image number number scene -> scene
; purpose: expects an image, an x coordinate, a y coordinate, and a scene,
;         and returns a new scene with that image centered at those
;         coordinates in the given scene

```

- **Example of a call to the `big-bang` function, in a `.rkt` file that includes definitions for functions `draw-penguin-scene` and `change-elevation`, which have the signatures:**

```

; signature: draw-penguin-scene: number -> scene

; signature: change-elevation: number string -> number

```

```

(big-bang 50
  (to-draw draw-penguin-scene)
  (on-tick add1)
  (on-key change-elevation)
  (stop-when zero?))

```