Practice Quiz 3

Due No due date **Points** 100 **Questions** 15 **Time limit** 50 Minutes
Allowed attempts Unlimited

Instructions

The instructions below are the exact ones you'll receive on the Quiz. You can ignore them for the Practice Quiz.

This exam is ONLY to be taken in Tech Auditorium during your scheduled exam time. Any other testing environment without explicit permission from the instructor will be considered a violation of our academic honesty policy.

- Please remember that you only get 1 attempt on this exam. You will have 50 minutes to complete it from the time that you start unless you have testing accommodations that allow for extra time.
- **Quiz Glossary (THIS LINK WORKS IN THE LOCKDOWN BROWSER) (https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)**

In part 1, if you get the answer *exactly* right, canvas will mark it as CORRECT automatically. If there is any slight variation it will mark it as INCORRECT. After the exam is over, we will be going through each question and doing manual grading. For example, if you write (list of str) and Canvas expected (listof string), canvas will automatically mark your answer wrong. However, when we go in to manually grade, we would mark the above answer as fully correct.

In part 2 and part 3, we will be grading the answers completely by hand. What we are looking for is that you understand what was wrong with the code and know how to fix it. We can utilize line number(s) as a reference point, so that you can indicate where the error occurred, but more important will be your rewrite. We will grade these with the same leniency and partial credit that we would in a printed exam.

If you encounter any issues while taking the exam, please raise your immediately!

Take the quiz again

Attempt history

	Attempt	Time	Score
LATEST	Attempt 1	50 minutes	0 out of 100 *

* Some questions not yet graded

Submitted 20 Nov at 13:13

Give the type of each of the following expressions. If more than one expression is given, assume that each one is executed, in order, and give the type of the value for the **last** expression. You may assume that all related libraries have been correctly loaded and that we are working in the **Advanced Student Language.**

- If it is a **primitive type** such as a number, string, Boolean or image (picture), just give the type.
- If it is a **record type**, just give the name of the record type. For example, if it's an album object, just say "album"
- If it is a **list**
  - If all the elements of the list are the same type, say "(listof *type*)" where *type* is the type of data in the list. For example (list 1 2 3) is a `(listof number)`.
  - If it is a with different types of data, say `(listof any)`
  - If you know the result is specifically the empty list, which has no elements and therefore no element type, just say empty list.
  - If you know the result is a **list** but you don't know the type of data in it, just say "list" and we will give you partial credit.
- If the result is a **function**, *give its argument and return types.* That is, write the type(s) of its argument(s) followed by an arrow and the type of its result. If the function accepts any type of value for an argument, just say "any". For example:
  - The type of the **sin** function is: `number -> number`
  - The type of the **integer?** function is: `any -> boolean`
  - The type of the **<** function is: `number number -> boolean`
  - The type of the **square** function is: `number string color -> image`
- If you know the expression's value is a **function** but don't know its argument or return types, just say **function**, and we will give you partial credit.
- If it returns **no value,** say "void"
- If executing it would produce an **exception**, say "exception." You don't need to specify the type of exception. This includes a program that will run out of memory due to infinite recursion.

[Quiz 3 Glossary (THIS LINK WORKS IN THE LOCKDOWN BROWSER)](https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)

Unanswered

Question 1

0 / 5 pts

```
(apply + (list 1 (list 2 3)))
```

You Answered

Correct Answers

exception

Unanswered

Question 2

0 / 5 pts

```
(define x true)
(when (not x)
  (+ 4 5 6))
```

You Answered

Correct Answers

void

(void)

Unanswered

Question 3

0 / 5 pts

```
(define sum 0)
(map (lambda (x)
```

```
      (set! sum (+ sum x)))
    (list 4 5 6))
```

ou Answered

orrect Answers

(listof void)

(listof (void))

list of voids

list of void

Unanswered

Question 4

0 / 5 pts

```
(define sum 0)
(begin (map (lambda (x)
              (set! sum (+ sum x)))
            (list 4 5 6))
       sum)
```

ou Answered

orrect Answers

num

number

15

integer

Unanswered

Question 5

0 / 5 pts

```
(cons (list true true) false)
```

ou Answered

orrect Answers

exception

Refer to the following code for the rest of the questions in Part 1.

```
; an animal is:
; - (make-animal string number number)
(define-struct animal (name weight age)
  #:methods
  (define (age-one-year! a)
    (begin (set-animal-age! a (+ 1 (animal-age a)))
           (animal-age a)))
  )

; a cat is:
; - (make-cat string number number string)
(define-struct (cat animal) (sleeping-spot)
  #:methods
  (define (feed! c)
    (begin (set-animal-weight! c (+ 2 (animal-weight c)))
           (animal-weight c)))
  )

; a dog is:
; - (make-dog string number number boolean)
(define-struct (dog animal) (athletic?)
  #:methods
  (define (feed! d)
    (begin (set-animal-weight! d (+ 3 (animal-weight d)))
           (animal-weight d)))
  )

(define pixel (make-cat "pixel" 12 15 "outside"))

(define gizmo (make-dog "gizmo" 12 2 #true))

(define myzoo (list (make-cat "buffer" 10 13 "couch")
                    (make-dog "lassie" 20 19 false)))
```

Unanswered

Question 6

0 / 5 pts

```
(age-one-year! gizmo)
```

ou Answered

orrect Answers

number

num

integer

int

Unanswered

Question 7

0 / 5 pts

```
(begin (feed! pixel)
       (cat-sleeping-spot pixel))
```

ou Answered

orrect Answers

string

str

Unanswered

Question 8

0 / 5 pts

```
set-cat-sleeping-spot!
```

ou Answered

orrect Answers

cat string -> void

cat, string -> void

cat string -> (void)

cat, string -> (void)

Unanswered

Question 9

0 / 5 pts

```
(dog-athletic? (second myzoo))
```

You Answered

Correct Answers

boolean

bool

Unanswered

Question 10

0 / 5 pts

```
(map feed! myzoo)
```

You Answered

Correct Answers

(listof number)

(listof numbers)

(list of numbers)

list of numbers

listof number

list of number

Each of the following questions shows some code being executed (with a line number on the left of each line) at the Racket prompt, along with the output or error it generated, and the intended output that the programmer wanted. Give the **correction** to the code to produce the desired result.

Remember, **you are not allowed to completely rewrite the given program.**

It is highly recommended that you copy and paste the code and then correct it as your answer. If you correctly identify the error and fix it you will get full credit. When copy and pasting, you can highlight just the code without the line numbers.

If you'd like your code to look more like Racket code, you can use the Canvas Toolbar to select the PREFORMATTED style. In the toolbar for the answer box, you'll see a button that says "Paragraph" with a down arrow (right next to the font size). Click on that then select Preformatted which will format your Text to look more like Racket code.

If you are on a Chromebook that doesn't allow Ctrl-C then Ctrl-V, you can highlight the code and then "drag it" to the box below in order to copy.

What we are looking for is that you understand what was wrong with the code and know how to fix it. If you are unsure of what the problem is, then you can provide an explanation of your thoughts. We will accept the line number(s) as a reference point, so that you can indicate where the error occurred, but more important will be your rewrite. We will grade these with the same leniency and partial credit that we would in a printed exam.

Quiz Glossary (THIS LINK WORKS IN THE LOCKDOWN BROWSER) (https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)

Refer to the following code for the first two questions in Part 2:

```
; an animal is:
; - (make-animal string number number)
(define-struct animal (name weight age)
  #:methods
  (define (age-one-year! a)
    (begin (set-animal-age! a (+ 1 (animal-age a)))
           (animal-age a)))
  )

; a cat is:
; - (make-cat string number number string)
(define-struct (cat animal) (sleeping-spot)
  #:methods
```

```
  (define (feed! c)
    (begin (set-animal-weight! c (+ 2 (animal-weight c)))
           (animal-weight c)))
  )

; a dog is:
; - (make-dog string number number boolean)
(define-struct (dog animal) (athletic?)
  #:methods
  (define (feed! d)
    (begin (set-animal-weight! d (+ 3 (animal-weight d)))
           (animal-weight d)))
  )

(define pixel (make-cat "pixel" 12 15 "outside"))

(define gizmo (make-dog "gizmo" 12 2 #true))

(define myzoo (list (make-cat "buffer" 10 13 "couch")
                    (make-dog "lassie" 20 19 false)))
```

Unanswered

Question 11

Not yet graded / 10 pts

Interaction

```
1  ; please refer to the animal, cat and dog struct definitions above.
2  (set-cat-weight! pixel 25)
```

Actual Output:

```
set-cat-weight!: this function is not defined
```

Desired Output:

No output desired. We would like to set the weight of pixel to 25.

Your answer:

needs to be a call to set-animal-weight!

cat doesn't have weight as a property but its parent type does

properties don't work the same as methods!

Unanswered

Question 12

Not yet graded / 10 pts

Interaction

```
(define mydogs
  (list (make-dog "gizmo" 12 2 #true)
        (make-dog "lassie" 20 19 #false)))

(define (print-athletic-dogs! lod)
  (begin (printf "these dogs are athletic: ")
         (for-each (lambda (d)
                     (when (dog-athletic? lod)
                           (printf (animal-name d))))
                   lod)))
(print-athletic-dogs! mydogs)
```

Actual Output:

```
dog-athletic?: expects a dog, given (list (make-dog "gizmo" 12 2 #true) (mak
e-dog "lassie" 20 19 #false))
```

Desired Output:

```
these dogs are athletic: gizmo
```

Your answer:

Need to modify the program to test a specific dog to be athletic!

line 8: (when (dog-athletic? d)

Unanswered

Question 13

Not yet graded / 10 pts

Interaction

```
(define (my-reverse lst)
  (local [(define rem lst)
          (define acc empty)
          (define (help)
            (cond [(empty? rem)  acc]
                  [else          (begin (cons (first rem) acc)
                                        (set! rem (rest rem))
                                        (help))]))]
         (help)))
(my-reverse (list 1 2 3))
```

Actual Output:

```
'() ; (an empty list)
```

Desired Output:

```
(list 3 2 1)
```

Your answer:

Need to set! the accumulator too!

line 6: ...(begin (set! acc (cons (first rem) acc)

Unanswered

Question 14

Not yet graded / 10 pts

Interaction

```
1  (define (multiply-all-and-sum lst factor)
2    (foldl +
3           0
4           (for-each (lambda (n)
5                       (* n factor))
6                     lst)))
7
8  (multiply-all-and-sum (list 1 2 3) 10)
```

Actual Output:

```
foldl: 3rd argument must be a list, given (void)
```

Desired Output:

```
60 ; (the result of 1*10 + 2*10 + 3*10)
```

Your answer:

for-each returns void

you want to use map instead

line 4: (map (lambda (n)

The following questions show a function definition. In the blank below you **may be asked to provide multiple tests** (`check-expect`) of the function. That is you, need to come up with and write valid tests that assesses whether or not the function works as the programmer intends **and that cover the specific cases referenced in the question**.

If you'd like your code to look more like Racket code, you can use the Canvas Toolbar to select the PREFORMATTED style. In the toolbar for

the answer box, you'll see a button that says "Paragraph" with a down arrow (right next to the font size). Click on that then select Preformatted which will format your Text to look more like Racket code.

[Quiz 3 Glossary (THIS LINK WORKS IN THE LOCKDOWN BROWSER) (https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)]

Unanswered

Question 15

Not yet graded / 10 pts

Consider the following atm code:

```
; deposit!: number -> number
; Deposits money into our bank account
; Effect: balance increases by deposit amount
(define deposit!
  (λ (amount)
    (begin (set! balance
                 (+ balance amount))
           balance)))

; withdraw!: number -> number
; Withdraws money from our account
; Effect: balance decreases by deposit amount unless
; balance is less than withdrawal amount
(define withdraw!
  (λ (amount)
    (begin (if (> amount balance)
               (error "Not enough money!")
               (set! balance (- balance amount)))
           balance)))
```

First write a definition of some global variable **balance** that starts at 1000 and then write a **series of at least 3 check expects** that check to see the following transactions are processed correctly according to the code above (in this exact order):

- A withdrawal of 700
- A deposit of 50
- A withdrawal of 1000

Hint: **check-error** allows you to check if you ran into a specific exception/error when running a function.

Your answer:

```
(define balance 1000)
(check-expect (withdraw! 700)
              300)
(check-expect (deposit! 50)
              350)
(check-error (withdraw! 1000)
             "Not enough money!")
```