

Practice Quiz 1b

Due No due date

Points 100


Questions 15

Time limit 50 Minutes

Allowed attempts Unlimited

Instructions

Please note that during the real exam, there will be a paper copy of the Quiz Glossary available but you can also just pull it up as a separate tab in the Lockdown Browser.

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

<These are the exact instructions that will be on the real Quiz! They're just provided for reference here and you can ignore them for the practice quizzes.>

- This exam is ONLY to be taken in Tech Auditorium. Please do not open this exam from any other location.
- 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.

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 will utilize the line number(s) as a reference point, so that you can indicate where the error occurred, but more important will be your rewrite. Feel free to provide a written explanation for what is wrong in the code and how you will fix it. 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 hand!

Take the quiz again

Attempt history

	Attempt	Time	Score
LATEST	Attempt 1	7 minutes	50 out of 100 *

* Some questions not yet graded

Submitted 16 Oct at 12:39

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.

- 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 (struct)**, 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 procedure 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 executing it would produce an **exception**, say "Exception." You don't need to specify the type of exception.

Question 1

5 / 5 pts

```
(foldr append
      (list)
      (list (list "a" "b")
            (list "c" "d")))
```

Correct!

(listof string)

Correct Answers

(listof strings)

(listof string)

listof string

Question 2

5 / 5 pts

```
(+ 1 1)
```

Correct!

number

Correct Answers

number

Question 3

5 / 5 pts

```
; You are given the following definitions:
(define-struct car (name num-seats))
```

```
(define a (make-car "mycar" 7))
```

```
; What is the type of value returned  
; for the following expression?  
(car-name a)
```

Correct!

string

Correct Answers

string

Question 4

5 / 5 pts

```
(append (list 1) (list 2))
```

Correct!

(listof number)

Correct Answers

(listof num)

(listof numbers)

(listof number)

Question 5

5 / 5 pts

```
(lambda (x y)  
  (if (= x 1)  
      (+ y 1)  
      10))
```

Correct!

number number -> number

Correct Answers

number number -> number

number, number -> number

Question 6

5 / 5 pts

```
(beside (square 10 "solid" "blue")  
        (circle 10 "solid" "purple"))
```

Correct!

image

Correct Answers

image

Question 7

5 / 5 pts

```
(lambda (y z)  
  (iterated-overlay (lambda (n)  
                      (circle n z "black"))  
                    y))
```

Correct!

number string -> image

Correct Answers

number string -> image

number, string -> image

Question 8

5 / 5 pts

```
(filter string-append (list "a" "b" "c"))
```

Correct!

exception

Correct Answers

exception

Question 9

5 / 5 pts

```
(local [(define a 5)]  
  (+ a 6))
```

Correct!

number

Correct Answers

number

Question 10

5 / 5 pts

```
; You are given the following definitions:  
(define-struct car (name num-seats))  
(define a (make-car "mycar" 7))  
  
; What is the type of value returned  
; for the following expression?  
(car? a)
```

Correct!

boolean

Correct Answers

boolean

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.

It is highly recommended that you copy and paste the code and then correct it as your answer (you can leave the line numbers in). If you correctly identify the error and fix it you will get full credit.

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.

Question 11

Not yet graded / 10 pts

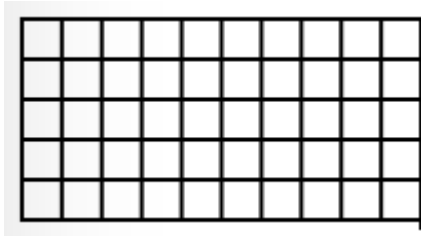
Interaction

```
1 (iterated-above (iterated-beside (square 10 "outline"  
2 "black")  
3 5) 10)
```

Actual Output:

```
iterated-beside: expects procedure, given #<image>
```

Desired Output:



Your answer:

```
1 (iterated-above (lambda (n) (iterated-beside (lambda (n) (square 10 "outli
2 ne" "black"))
3                                     10))
                                     5)
```

```
(iterated-above (lambda (n)
                  (iterated-beside (lambda (m)
                                     (square 10 "outline" "black"))
                                     10))
                  5)
```

Question 12

Not yet graded / 10 pts

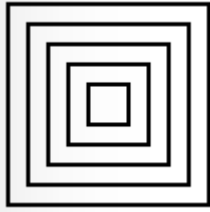
Interaction

```
1 (overlay (map (lambda (size)
2               (square size "outline" "black")))
3           (list 10 20 30 40 50)))
```

Actual Output:

```
overlay: expects at least 2 arguments, but found only 1
```

Desired Output:



Your answer:

```
1 (apply overlay (map (λ (size)
2                     (square size "outline" "black")))
3                     (list 10 20 30 40 50)))
```

```
(apply overlay
  (map (λ (size)
        (square size "outline" "black")))
  (list 10 20 30 40 50)))
```

Overlay is supposed to take a series of pictures as arguments. The programmer had instead been passing it a single list of pictures as its argument. You need to use `apply`, which pulls the items out of the list and passes them as separate arguments to `overlay`.

Question 13

Not yet graded / 10 pts

Interaction

```
1 (foldl +
2     (map (λ (n) (* n n))
3         (list 1 2 3 4 5)))
```

Actual Output:

```
foldl: expects 3 arguments, but found only 2
```

Desired Output:

55 (the sum of the squares of the elements of the list).

Your answer:

```
1 (foldl + 0
2       (map (λ (n) (* n n))
3       (list 1 2 3 4 5)))
```

```
(foldl +
  0
  (map (λ (n) (* n n))
    (list 1 2 3 4 5)))
```

The programmer forgot to include the second argument to foldl.

Question 14

Not yet graded / 10 pts

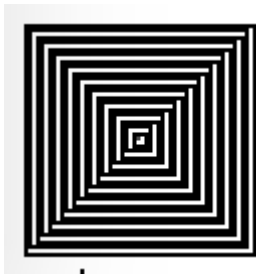
Interaction

```
1 (iterated-overlay λ (n) (square (* 3 n) "outline" "black"
2                               20)
```

Actual Output:

```
lambda: expected an open parenthesis before lambda, but found none
```

Desired Output:



Your answer:

```
1 (iterated-overlay (λ (n) (square (* 3 n) "outline" "black"))
2                      20)
```

```
(iterated-overlay (λ (n) (square (* 3 n) "outline" "black"))
                  20)
```

The following questions show a function definition. In the blank below, please provide one valid test (`check-expect`) of the procedure.

Question 15

Not yet graded / 10 pts

```
; sum: (listof number) -> number
; returns the sum of the items in the list
(define sum (lambda (the-list)
  (foldl + 0 the-list)))
```

In the blank below, please provide one valid test (`check-expect`) of the procedure.

Your answer:

```
(check-expect (sum (list 1 2 3 4)) 10)
```

One possible answer.

```
(check-expect (sum (list 1 2 3 4)) 10)
```