

# Racket Programming Assignment #3: Lambda and Basic Lisp

## **Learning Abstract:**

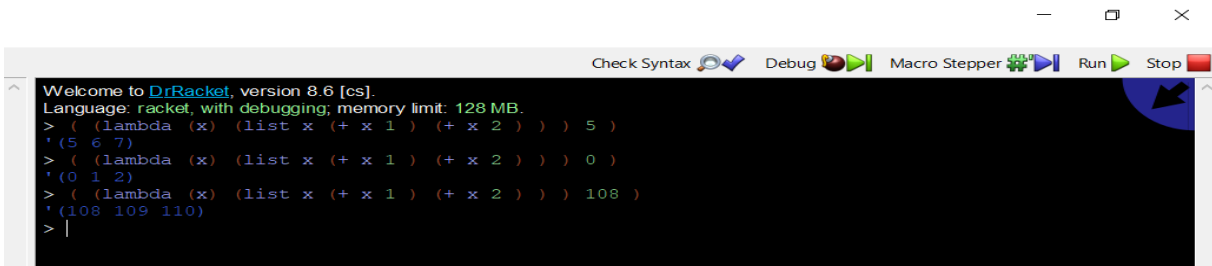
The first assignment required using small undefined anonymous lambda functions and mimicking the demos that were provided

The second assignment required us to produce the output for the functions we were given in lesson 6

The third assignment required us to recreate a program called sampler and use that code as a base for a program called color thing. The sampler program would take input and then output a random element from the list that was just inputted. The color program used our sampler code for its base, but this program would select a color from the given list and then output a rectangle of a specified size with that color.

The fourth assignment required us to create several programs for poker. We were given a initial piece of code and was asked to modify it as we proceeded through the tasks. The functions that would be added were to pick two cards, determine a higher rank, classify two cards, and classify two cards while determining which is higher.

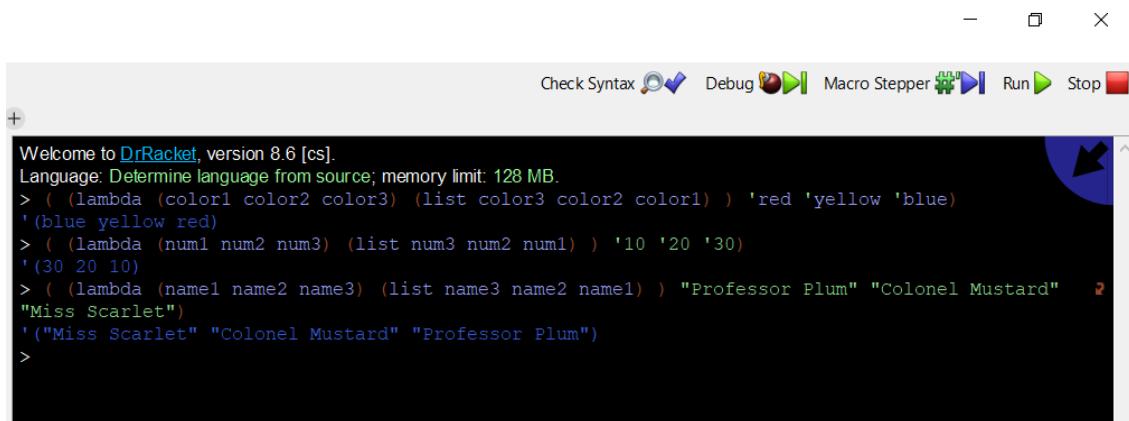
# Task 1 Lambda:



A screenshot of the DrRacket IDE interface. The title bar shows 'Check Syntax', 'Debug', 'Macro Stepper', 'Run', and 'Stop' buttons. The main text area contains the following Racket code:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( (lambda (x) (list x (+ x 1) (+ x 2) ) ) 5 )
'(5 6 7)
> ( (lambda (x) (list x (+ x 1) (+ x 2) ) ) 0 )
'(0 1 2)
> ( (lambda (x) (list x (+ x 1) (+ x 2) ) ) 108 )
'(108 109 110)
> |
```

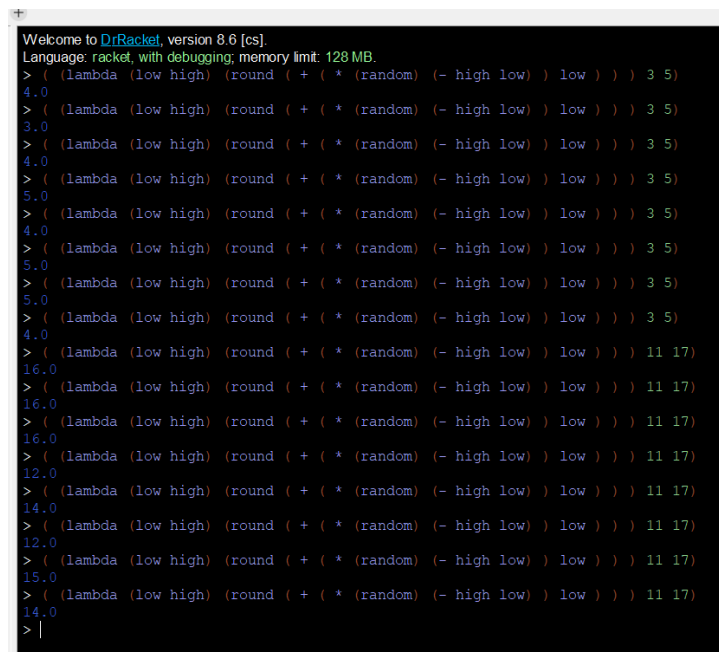
Task 1a code



A screenshot of the DrRacket IDE interface. The title bar shows 'Check Syntax', 'Debug', 'Macro Stepper', 'Run', and 'Stop' buttons. The main text area contains the following Racket code:

```
Welcome to DrRacket, version 8.6 [cs].
Language: Determine language from source; memory limit: 128 MB.
> ( (lambda (color1 color2 color3) (list color3 color2 color1) ) 'red 'yellow 'blue)
'(blue yellow red)
> ( (lambda (num1 num2 num3) (list num3 num2 num1) ) '10 '20 '30)
'(30 20 10)
> ( (lambda (name1 name2 name3) (list name3 name2 name1) ) "Professor Plum" "Colonel Mustard"
"Miss Scarlet")
'("Miss Scarlet" "Colonel Mustard" "Professor Plum")
> |
```

Task 1b code



A screenshot of the DrRacket IDE interface. The title bar shows 'Check Syntax', 'Debug', 'Macro Stepper', 'Run', and 'Stop' buttons. The main text area contains the following Racket code:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
4.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
3.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
4.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
5.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
4.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
5.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
4.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 3 5)
16.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
16.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
16.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
12.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
14.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
12.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
15.0
> ( (lambda (low high) (round (+ (* (random) (- high low) ) low ) ) ) 11 17)
14.0
> |
```

Task 1c code

# Task 2 List Processing References and Constructors

```
+
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define colors '(red blue yellow orange) )
> colors
'(red blue yellow orange)
> 'colors
'colors
> (quote colors)
'colors
> (car colors)
'red
> (cdr colors)
'(blue yellow orange)
> (car (cdr colors) )
'blue
> (cdr (cdr colors) )
'(yellow orange)
> (cadr colors)
'blue
> (caddr colors)
'(yellow orange)
> (first colors)
'red
> (second colors)
'blue
> (third colors)
'yellow
> (list-ref colors 2 )
'yellow
>
```

Task 2 part 1

```
+
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define key-of-c '(c d e) )
> (define key-of-g '(g a b) )
> (cons key-of-c key-of-g)
'((c d e) g a b)
> (list key-of-c key-of-g)
'((c d e) (g a b))
> (append key-of-c key-of-g)
'(c d e g a b)
> (define pitches '(do re mi fa so la ti))
> (car (cdr (cdr (cdr animals) )))
animals: undefined;
cannot reference an identifier before its definition
> (caddr pitches)
'fa
> (list-ref pitches 3)
'fa
> (define a 'alligator)
> (define b 'pussycat)
> (define c 'chimpanzee)
> (cons a (cons b (cons c '())))
'(alligator pussycat chimpanzee)
> (list a b c)
'(alligator pussycat chimpanzee)
> (define x '(1 one))
> (define y '(2 two))
> (cons (car x) (cons (car (cdr x)) y))
'(1 one 2 two)
> (append x y)
'(1 one 2 two)
> |
```

Task 2 Part 2

# Task 3 Little Color Interpreter

```
#lang racket
(define (sampler)
  (display "(?): " ))
  (define the-list (read))
  (define the-element
    (list-ref the-list (random (length the-list))))
  (display the-element) (display "\n")
  (sampler))

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging, memory limit: 128 MB.
> (sampler)
(?): (red orange green blue yellow indigo violet)
orange
(?): (red orange green blue yellow indigo violet)
indigo
(?): (red orange green blue yellow indigo violet)
indigo
(?): (red orange green blue yellow indigo violet)
red
(?): (red orange green blue yellow indigo violet)
indigo
(?): (aet ate eat eta tae tea)
ate
(?): (aet ate eat eta tae tea)
tea
(?): (aet ate eat eta tae tea)
tea
(?): (aet ate eat eta tae tea)
ate
(?): (aet ate eat eta tae tea)
eat
(?): (0 1 2 3 4 5 6 7 8 9)
6
(?): (0 1 2 3 4 5 6 7 8 9)
2
(?): (0 1 2 3 4 5 6 7 8 9)
4
(?): (0 1 2 3 4 5 6 7 8 9)
9
(?): (0 1 2 3 4 5 6 7 8 9)
8
(?): ..user break
length: contract violation
expected: list?
given: '..user
> |
```

Task 3A code and demo

```
1 #lang racket
2 (require 2htdp/image)
3
4 (define (color-thing)
5   (display "(?): ")
6   (define the-list (read))
7   (define the-choice (car the-list))
8   (define color-list (cadr the-list))
9   (define (random-color)
10    (display (rectangle 400 25 "solid" (list-ref color-list (random (length color-list)))))
11    (define (select-color)
12      (display (rectangle 400 25 "solid" (list-ref color-list (- the-choice 1))))
13      (define (every-color)
14        (display (color-maker 0)))
15        (define (color-maker n)
16          (cond
17            [(= n (- (length color-list) 1))
18              (display
19                (rectangle 400 25 "solid"
20                  (list-ref color-list n)))
21              (display "\n")
22              (color-thing))
19            [else
23              (display
24                (rectangle 400 25 "solid"
25                  (list-ref color-list n)))
26              (display "\n")
27              (color-maker (+ n 1)))]))
28          (cond
29            [(equal? the-choice 'random)
30              (random-color)]
31            [(equal? the-choice 'all)
32              (every-color)]
33            [else
34              (select-color)]))
35          (display "\n")
36          (color-thing))
37
38 (color-thing))

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging, memory limit: 128 MB.
> (color-thing)
(?): (random (olivedrab dodgerblue indigo plum teal darkorange))
(?): (random (olivedrab dodgerblue indigo plum teal darkorange))
(?): (random (olivedrab dodgerblue indigo plum teal darkorange))
(?): (all (olivedrab dodgerblue indigo plum teal darkorange))
(?): (2 (olivedrab dodgerblue indigo plum teal darkorange))
(?): (3 (olivedrab dodgerblue indigo plum teal darkorange))
(?): (5 (olivedrab dodgerblue indigo plum teal darkorange))
(?): break
caz: contract violation
expected: pair?
given: 'break
>
```

Task 3B code and demo 1

```

1 #lang racket
2 (require 2htdp/image)
3
4 (define (color-thing)
5   (display "[?]: ")
6   (define the-list (read))
7   (define the-choice (car the-list))
8   (define color-list (cadr the-list))
9   (define (random-color)
10    (display (rectangle 400 25 "solid" (list-ref color-list (random (length color-list))))
11    (define (select-color)
12    (display (rectangle 400 25 "solid" (list-ref color-list (- the-choice 1))))
13    (define (every-color)
14    (display (color-maker 0))
15    (define (color-maker n)
16    (cond
17      [(= n (- (length color-list) 1))
18       (display
19        (rectangle 400 25 "solid"
20         (list-ref color-list n))
21        (display "\n" )
22        (color-thing) )
23       (else
24        (display
25         (rectangle 400 25 "solid"
26          (list-ref color-list n))
27         (display "\n" )
28         (color-maker (+ n 1)))
29        (cond
30          [(equal? the-choice 'random)
31           (random-color)]
32          [(equal? the-choice 'all)
33           (every-color)]
34          (else
35           (select-color))
36          (display "\n" )

```

Welcome to DrRacket, version 8.6 [cs].  
Language: racket, with debugging, memory limit: 128 MB.

```

> (color-thing)
[?]: (random (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (random (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (random (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (all (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (2 (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (3 (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]: (6 (Crimson Chocolate Gold LimeGreen RoyalBlue Purple VioletRed) )
[?]:

```

Task 3B code and Demo 2

# Task 4 Two Card Poker

task4.rkt - DrRacket

File Edit View Language Racket Insert Scripts Tabs Help

task4.rkt (define ...)

```

1 #lang racket
2 (define (ranks rank)
3   (list
4     (list rank 'C)
5     (list rank 'D)
6     (list rank 'H)
7     (list rank 'S) ) )
8
9 (define (deck)
10  (append
11    (ranks 2)
12    (ranks 3)
13    (ranks 4)
14    (ranks 5)
15    (ranks 6)
16    (ranks 7)
17    (ranks 8)
18    (ranks 9)
19    (ranks 10)
20    (ranks 'K)
21    (ranks 'Q)
22    (ranks 'J)
23    (ranks 'A) ) )
24
25 (define (pick-a-card)
26   (define cards (deck))
27   (list-ref cards (random (length cards))))
28
29 (define (show card)
30   (display (rank card))
31   (display (suit card)))
32
33 (define (rank card)
34   (car card))
35
36 (define (suit card)
37   (cadr card))
38
39 (define (red? card)
40   (or
41     (equal? (suit card) 'D)
42     (equal? (suit card) 'H) ) )
43
44 (define (black? card)
45   (not (red? card)))

```

Determine language from source

```

1 #lang racket
43
44 (define (black? card)
45   (not (red? card) ) )
46
47 (define (aces? card1 card2 )
48   (and
49     (equal? (rank card1) 'A )
50     (equal? (rank card2) 'A ) ) )
51
52 (define (pick-two-cards)
53   (define c1 (pick-a-card) )
54   (define c2 (pick-a-card) )
55   (cond
56     ((equal? c1 c2)
57      (pick-two-cards) )
58     (else
59      (list c1 c2) ) ) )
60
61 (define (higher-rank c1 c2)
62   (display (list c1 c2) )
63   (display "\n")
64   (cond
65     ((equal? (rank c1) '10)
66      (display (rank c1) ) )
67     ((equal? (rank c2) '10)
68      (display (rank c2) ) )
69
70     ((equal? (rank c1) 'J)
71      (display (rank c1) ) )
72     ((equal? (rank c2) 'J)
73      (display (rank c2) ) )
74
75     ((equal? (rank c1) 'Q)
76      (display (rank c1) ) )
77     ((equal? (rank c2) 'Q)
78      (display (rank c2) ) )
79
80     ((equal? (rank c1) 'K)
81      (display (rank c1) ) )
82     ((equal? (rank c2) 'K)
83      (display (rank c2) ) )
84
85     ((equal? (rank c1) 'A)
86      (display (rank c1) ) )

```

Determine language from source

Type here to search

```

1 #lang racket
84
85 ((equal? (rank c1) 'A)
86  (display (rank c1) ) )
87 ((equal? (rank c2) 'A)
88  (display (rank c2) ) )
89 (else
90  (cond
91    ((equal? (rank c1) (rank c2))
92     (rank c1) )
93    ((> (rank c1) (rank c2) )
94     (display (rank c1) ) )
95    ((< (rank c1) (rank c2) )
96     (display (rank c2) ) ) ) ) )
97
98 (define (classify-two-cards-ur list-of-cards)
99   (display list-of-cards) (display ": ")
100   (define c1 (first list-of-cards) )
101   (define c2 (second list-of-cards) )
102   (cond
103     ((equal? (rank c1) (rank c2))
104      (display "Pair of ") (display (rank c1) ) (display"s" ) )
105     ((equal? (rank c1) '10)
106      (display (rank c1)) (display " High ") )
107     ((equal? (rank c2) '10)
108      (display (rank c2)) (display " High ") )
109
110     ((equal? (rank c1) 'J)
111      (display (rank c1)) (display " High ") )
112     ((equal? (rank c2) 'J)
113      (display (rank c2)) (display " High ") )
114
115     ((equal? (rank c1) 'Q)
116      (display (rank c1)) (display " High ") )
117     ((equal? (rank c2) 'Q)
118      (display (rank c2)) (display " High ") )
119
120     ((equal? (rank c1) 'K)
121      (display (rank c1)) (display " High ") )
122     ((equal? (rank c2) 'K)
123      (display (rank c2)) (display " High ") )
124
125     ((equal? (rank c1) 'A)
126      (display (rank c1)) (display " High ") )
127     ((equal? (rank c2) 'A)

```

Determine language from source

Type here to search

task4.rkt - DrRacket

File Edit View Language Racket Insert Scripts Tabs Help

task4.rkt (define...)

```
1 #lang racket
124
125 ((equal? (rank c1) 'A)
126 (display (rank c1)) (display " High ") )
127 ((equal? (rank c2) 'A)
128 (display (rank c2)) (display " High ") )
129 (else
130 (cond
131 ((> (rank c1) (rank c2) )
132 (display (rank c1)) (display " High ") )
133 ((< (rank c1) (rank c2))
134 (display (rank c2)) (display " High ") ) ) ) )
135
136 (straight? c1 c2)
137 (flush? c1 c2) )
138
139 (define (flush? c1 c2)
140 (cond
141 ((equal? (suit c1) (suit c2))
142 (display "Flush " ) ) )
143
144 (define (straight? c1 c2)
145 (cond
146 ((equal?(rank c1)'A)
147 (cond
148 ((equal?(rank c2)'K)
149 (display "Straight " ) ) )
150
151 ((equal?(rank c1) 'K)
152 (cond
153 ((equal?(rank c2)'A)
154 (display "Straight" )
155 ((equal?(rank c2)'Q)
156 (display "Straight" ) ) ) )
157
158 ((equal?(rank c1) 'Q)
159 (cond
160 ((equal?(rank c2)'K)
161 (display "Straight" )
162 ((equal?(rank c2)'J)
163 (display "Straight" ) ) ) )
164
165 ((equal?(rank c1) 'J)
166 (cond
167 ((equal?(rank c2)'10)
```

Determine language from source



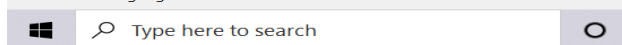
task4.rkt - DrRacket

File Edit View Language Racket Insert Scripts Tabs Help

task4.rkt (define...)

```
1 #lang racket
176
177 ((equal?(rank c1) 'J)
178 (cond
179 ((equal?(rank c2)'Q)
180 (display "Straight" )
181 ((equal?(rank c2)'10)
182 (display "Straight" ) ) ) )
183
184 ((equal?(rank c1) '10)
185 (cond
186 ((equal?(rank c2)'J)
187 (display "Straight" )
188 ((equal?(rank c2)'9)
189 (display "Straight" ) ) ) )
190
191 ((equal? (rank c1)'9)
192 (cond
193 ((equal? (rank c2) '10)
194 (display "Stright" )
195 (equal? (rank c2) '8)
196 (display "Straight" ) ) ) )
197
198 ((equal? (rank c1) '8)
199 (cond
200 ((equal? (rank c2)'9)
201 (display "Straight" )
202 ((equal? (rank c2) '7)
203 (display "Straight" ) ) ) )
204
205 ((equal? (rank c1) '7)
206 (cond
207 ((equal? (rank c2)'8)
208 (display "Straight" )
209 ((equal? (rank c2) '6)
210 (display "Straight" ) ) ) )
211
212 ((equal? (rank c1) '6)
213 (cond
214 ((equal? (rank c2)'7)
215 (display "Straight" )
216 ((equal? (rank c2) '5)
217 (display "Straight" ) ) ) )
218
219 ((equal? (rank c1) '5)
```

Determine language from source



```

1 #lang racket
210 (display "Straight" ) ) )
211
212 ((equal? (rank c1) '6)
213 (cond
214 ((equal? (rank c2) '7)
215 (display "Straight" )
216 ((equal? (rank c2) '5)
217 (display "Straight" ) ) ) )
218
219 ((equal? (rank c1) '5)
220 (cond
221 ((equal? (rank c2) '6)
222 (display "Straight" )
223 ((equal? (rank c2) '4)
224 (display "Straight" ) ) ) )
225
226 ((equal? (rank c1) '4)
227 (cond
228 ((equal? (rank c2) '5)
229 (display "Straight" )
230 ((equal? (rank c2) '3)
231 (display "Straight" ) ) ) )
232
233 ((equal? (rank c1) '3)
234 (cond
235 ((equal? (rank c2) '4)
236 (display "Straight" )
237 ((equal? (rank c2) '2)
238 (display "Straight" ) ) ) )
239
240 ((equal? (rank c1) '2)
241 (cond
242 ((equal? (rank c2) '3)
243 (display "Straight" )
244 ((equal? (rank c2) '1)
245 (display "Straight" ) ) ) )
246
247 ((equal? (rank c1) '1)
248 (cond
249 ((equal? (rank c2) '2)
250 (display "Straight" ) ) ) ) )
251
252 ; ( trace higher-rank )

```

Determine language from source

```

253
254 ;classify two cards
255 (define (classify-two-cards-r list-of-cards)
256 (display list-of-cards)(display ": ")
257 (define c1 (first list-of-cards) )
258 (define c2 (second list-of-cards) )
259 (cond
260 ((equal? (rank c1) (rank c2))
261 (display "Pair of " ) (display (rank c1) ) (display"s" ) )
262 ((equal? (rank c1) '1)
263 (display "One high" )
264 ((equal? (rank c2) '1)
265 (display "One high" ) )
266
267 ((equal? (rank c1) '2)
268 (display "Two high" )
269 ((equal? (rank c2) '2)
270 (display "Two high" ) )
271
272 ((equal? (rank c1) '3)
273 (display "Three high" )
274 ((equal? (rank c2) '3)
275 (display "Three high" ) )
276
277 ((equal? (rank c1) '4)
278 (display "Four high" )
279 ((equal? (rank c2) '4)
280 (display "four high" ) )
281
282 ((equal? (rank c1) '5)
283 (display "five high" )
284 ((equal? (rank c2) '5)
285 (display "five high" ) )
286
287 ((equal? (rank c1) '6)
288 (display "six high" )
289 ((equal? (rank c2) '6)
290 (display "six high" ) )
291
292 ((equal? (rank c1) '7)
293 (display "seven high" )
294 ((equal? (rank c2) '7)
295 (display "seven high" ) )
296

```



```
File Edit View Language Racket Insert Scripts Tabs Help
task4.rkt (define ...)
1 #lang racket
294 (equal? (rank c2) '7)
295 (display "seven high" )
296
297 (equal? (rank c1) '8)
298 (display "eight high" )
299 (equal? (rank c2) '8)
300 (display "eight high" )
301
302 (equal? (rank c1) '9)
303 (display "nine high" )
304 (equal? (rank c2) '9)
305 (display "nine high" )
306
307 (equal? (rank c1) '10)
308 (display "ten high" )
309 (equal? (rank c2) '10)
310 (display "ten high" )
311
312 (equal? (rank c1) 'J)
313 (display "Jack high" )
314 (equal? (rank c2) 'J)
315 (display "Jack high" )
316
317 (equal? (rank c1) 'Q)
318 (display "Queen high" )
319 (equal? (rank c2) 'Q)
320 (display "Queen high" )
321
322 (equal? (rank c1) 'K)
323 (display "King high" )
324 (equal? (rank c2) 'K)
325 (display "King high" )
326
327 (equal? (rank c1) 'A)
328 (display "Ace high" )
329 (equal? (rank c2) 'A)
330 (display "Ace high" )
331
332 (straight? c1 c2)
333 (flush? c1 c2)
334
335
336 ;( trace higher-rank )

Determine language from source
Type here to search
```

These pictures contain the code for task 4A, 4B and 4C

```
> (define c1 '(7 C) )
> (define c2 '(Q H) )
> c1
'(7 C)
> c2
'(Q H)
> (rank c1)
7
> (rank c2)
'Q
> (suit c1)
'C
> (suit c2)
'H
> (red? c1)
#f
> (red? c2)
#f
> (black? c1)
#t
> (black? c2)
#f
> (aces? '(A C) '(A S) )
#f
> (aces? '(K S) '(A C) )
#f
> (ranks 4)
'((4 C) (4 D) (4 H) (4 S))
> (ranks 'K)
'((K C) (K D) (K H) (K S))
> (length (deck) )
52
> (display (deck))
((2 C) (2 D) (2 H) (2 S) (3 C) (3 D) (3 H) (3 S) (4 C) (4 D) (4 H) (4 S) (5 C) (5 D) (5 H) (5 S) (6 C) (6 D) (6 H) (6 S) (7 C) (7 D) (7 H) (7 S) (8 C) (8 D) (8 H) (8 S) (9 C) (9 D) (9 H) (9 S) (10 C) (10 D) (10 H) (10 S) (K C) (K D) (K H) (K S) (Q C) (Q D) (Q H) (Q S) (J C) (J D) (J H) (J S) (A C) (A D) (A H) (A S))
> (pick-a-card)
'(9 C)
> (pick-a-card)
'(7 S)
> (pick-a-card)
'(J D)
> (pick-a-card)
'(7 S)
```

Task 4A Demo

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging, memory limit: 128 MB.
> (pick-two-cards)
'((3 S) (8 C))
> (pick-two-cards)
'((7 S) (5 S))
> (pick-two-cards)
'((7 D) (8 H))
> (pick-two-cards)
'((7 C) (8 C))
> (pick-two-cards)
'((7 C) (9 H))
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(K S) '(3 S))
((K S) (3 S))
K<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(A D) '(2 D))
((A D) (2 D))
A<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(4 S) '(7 C))
((4 S) (7 C))
7<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(10 C) '(5 C))
((10 C) (5 C))
10<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(6 H) '(7 C))
((6 H) (7 C))
7<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(J D) '(Q H))
((J D) (Q H))
J<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(Q S) '(10 C))
((Q S) (10 C))
10<#<void>
> (higher-rank (pick-a-card) (pick-a-card) )
>(higher-rank '(7 C) '(Q D))
((7 C) (Q D))
Q<#<void>
> |
```

Higher Rank demo

Task 4B Pick two cards and

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging, memory limit: 128 MB.
> (classify-two-cards-ur (pick-two-cards) )
((J H) (Q S)): J High
> (classify-two-cards-ur (pick-two-cards) )
((2 D) (K S)): K High
> (classify-two-cards-ur (pick-two-cards) )
((A D) (4 D)): A High Flush
> (classify-two-cards-ur (pick-two-cards) )
((K S) (Q S)): Q High Straight Flush
> (classify-two-cards-ur (pick-two-cards) )
((6 C) (Q C)): Q High Flush
> (classify-two-cards-ur (pick-two-cards) )
((6 C) (8 S)): 8 High
> (classify-two-cards-ur (pick-two-cards) )
((K C) (A C)): K High Straight Flush
> (classify-two-cards-ur (pick-two-cards) )
((3 D) (J C)): J High
> (classify-two-cards-ur (pick-two-cards) )
((Q S) (2 D)): Q High
> (classify-two-cards-ur (pick-two-cards) )
((5 H) (A S)): A High
> (classify-two-cards-ur (pick-two-cards) )
((10 S) (4 S)): 10 High Flush
> (classify-two-cards-ur (pick-two-cards) )
((6 S) (Q S)): Q High Flush
> (classify-two-cards-ur (pick-two-cards) )
((Q D) (7 C)): Q High
> (classify-two-cards-ur (pick-two-cards) )
((6 D) (8 D)): 8 High Flush
> (classify-two-cards-ur (pick-two-cards) )
((10 S) (7 C)): 10 High
> (classify-two-cards-ur (pick-two-cards) )
((A H) (5 C)): A High
> (classify-two-cards-ur (pick-two-cards) )
((5 H) (9 C)): 9 High
> (classify-two-cards-ur (pick-two-cards) )
((6 C) (5 C)): 6 High Straight Flush
> (classify-two-cards-ur (pick-two-cards) )
((9 S) (7 C)): 9 High
> (classify-two-cards-ur (pick-two-cards) )
((8 H) (5 C)): 8 High
>
```

Task 4B Classify-two-cards-ur

```

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (classify-two-cards-r (pick-two-cards) )
((K H) (3 C)): Three high
> (classify-two-cards-r (pick-two-cards) )
((7 C) (A C)): seven highFlush
> (classify-two-cards-r (pick-two-cards) )
((Q C) (8 C)): eight highFlush
> (classify-two-cards-r (pick-two-cards) )
((9 H) (A S)): nine high
> (classify-two-cards-r (pick-two-cards) )
((5 C) (7 H)): five high
> (classify-two-cards-r (pick-two-cards) )
((2 S) (9 S)): Two highFlush
> (classify-two-cards-r (pick-two-cards) )
((2 C) (K D)): Two high
> (classify-two-cards-r (pick-two-cards) )
((J S) (A D)): Jack highStraight
> (classify-two-cards-r (pick-two-cards) )
((A H) (2 H)): Two highFlush
> (classify-two-cards-r (pick-two-cards) )
((Q D) (A D)): Queen highFlush
> (classify-two-cards-r (pick-two-cards) )
((10 S) (4 C)): four high
> (classify-two-cards-r (pick-two-cards) )
((7 H) (4 H)): four highFlush
> (classify-two-cards-r (pick-two-cards) )
((Q D) (8 C)): eight high
> (classify-two-cards-r (pick-two-cards) )
((3 C) (7 S)): Three high
> (classify-two-cards-r (pick-two-cards) )
((J C) (Q D)): Jack high
> (classify-two-cards-r (pick-two-cards) )
((7 S) (2 S)): Two highFlush
> (classify-two-cards-r (pick-two-cards) )
((2 S) (3 C)): Two highStraight
> (classify-two-cards-r (pick-two-cards) )
((Q S) (7 C)): seven high
> (classify-two-cards-r (pick-two-cards) )
((J D) (Q H)): Jack high
> (classify-two-cards-r (pick-two-cards) )
((2 D) (7 S)): Two high
> (classify-two-cards-r (pick-two-cards) )
((6 D) (A D)): six highFlush
>

```

Task 4C Classifier demo