Temitope Emokpae 10/7/2022

CSC 344

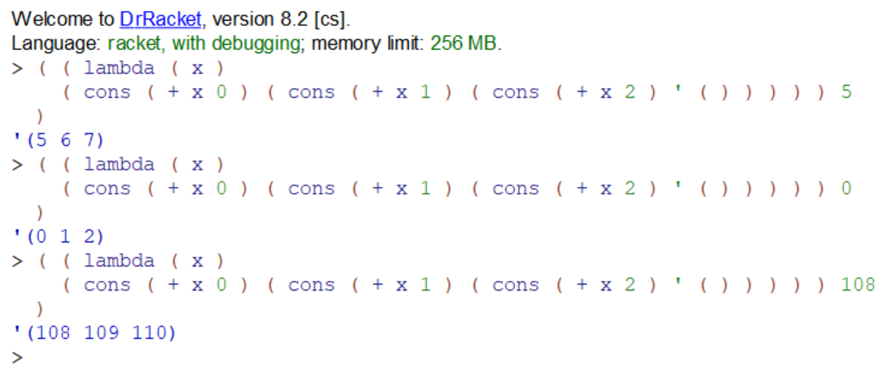
Racket Programming Assignment #3: Lambda and Basic Lisp

Learning Abstract

This assignment features how historical lisp works. The first task focuses on the lambda function. The second task focuses on list references and constructors based on the demo in Lesson 6 - “Basics Lisp Programming”. The third task focuses on a “sampler” code as well as a colors-thing coded based on the sampler code and mimicking the demos found in the same Lesson 6. And the last one task focuses on a Two Card Poker program and mimicking the demon found in the same Lesson 6.

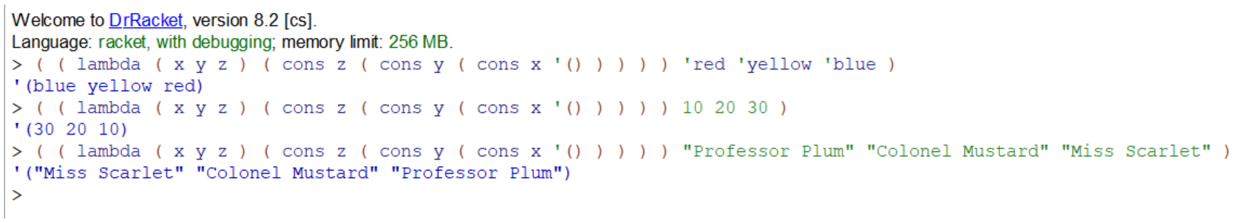
Problem 1: Lambda

Problem 1a - Three ascending integers



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( ( lambda ( x )
    ( cons ( + x 0 ) ( cons ( + x 1 ) ( cons ( + x 2 ) ' ( ) ) ) ) ) 5
  )
'(5 6 7)
> ( ( lambda ( x )
    ( cons ( + x 0 ) ( cons ( + x 1 ) ( cons ( + x 2 ) ' ( ) ) ) ) ) 0
  )
'(0 1 2)
> ( ( lambda ( x )
    ( cons ( + x 0 ) ( cons ( + x 1 ) ( cons ( + x 2 ) ' ( ) ) ) ) ) 108
  )
'(108 109 110)
>
```

Problem 1b - Make list in reverse order



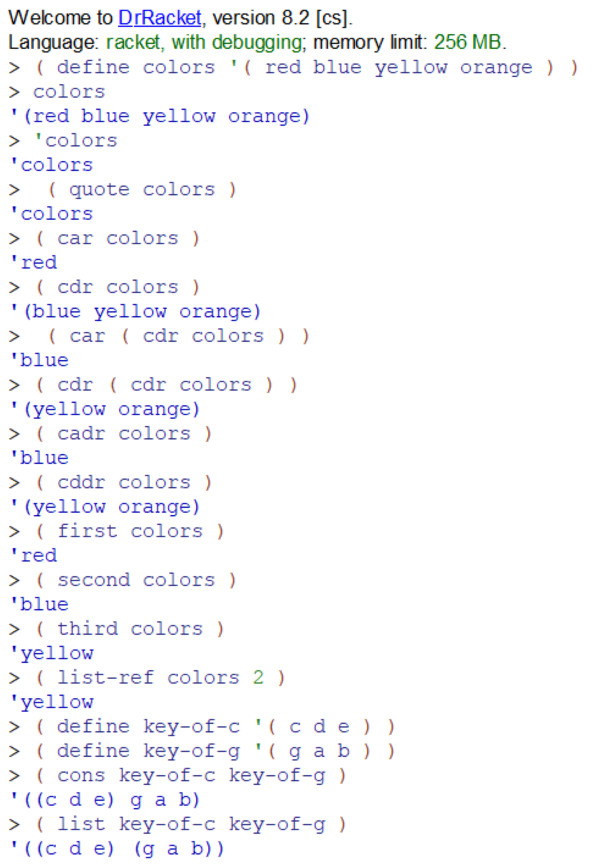
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( ( lambda ( x y z ) ( cons z ( cons y ( cons x '() ) ) ) ) 'red 'yellow 'blue )
'(blue yellow red)
> ( ( lambda ( x y z ) ( cons z ( cons y ( cons x '() ) ) ) ) 10 20 30 )
'(30 20 10)
> ( ( lambda ( x y z ) ( cons z ( cons y ( cons x '() ) ) ) ) "Professor Plum" "Colonel Mustard" "Miss Scarlet" )
'("Miss Scarlet" "Colonel Mustard" "Professor Plum")
>
```

Problem 1c - Random number generator

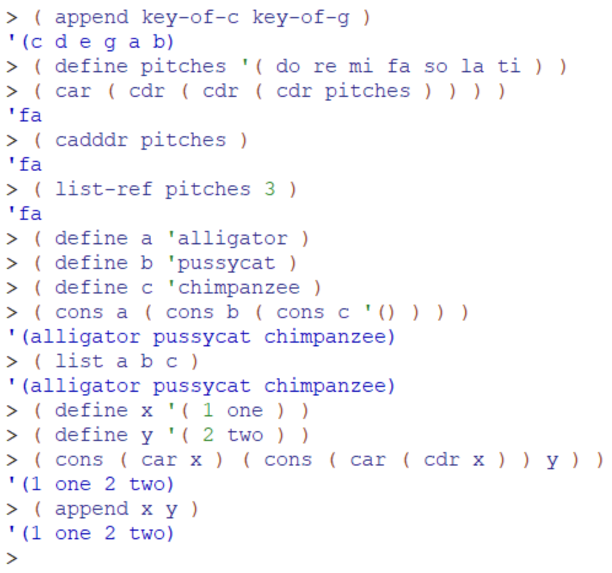
```
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
3
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
3
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
4
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
5
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 3 5 )
3
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
11
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
12
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
11
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
17
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
15
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
12
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
13
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
13
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
17
> ( ( lambda ( x y ) ( + ( random ( - ( + y 1 ) x ) ) x ) ) 11 17 )
13
```

Problem 2: List Processing Referencers and Constructors

Demo



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 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
> ( cddr colors )
'(yellow orange)
> ( first colors )
'red
> ( second colors )
'blue
> ( third colors )
'yellow
> ( list-ref colors 2 )
'yellow
> ( 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 pitches ) ) ) )
'fa
> ( cadddr 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)
>
```

Problem 3: Little Color Interpreter

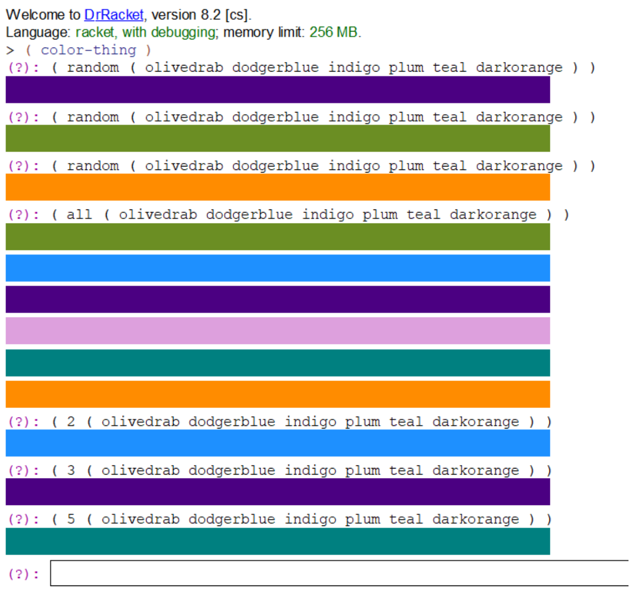
Problem 3a - Establishing the Sampler code from Lesson 6

```
#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 )
   )
```

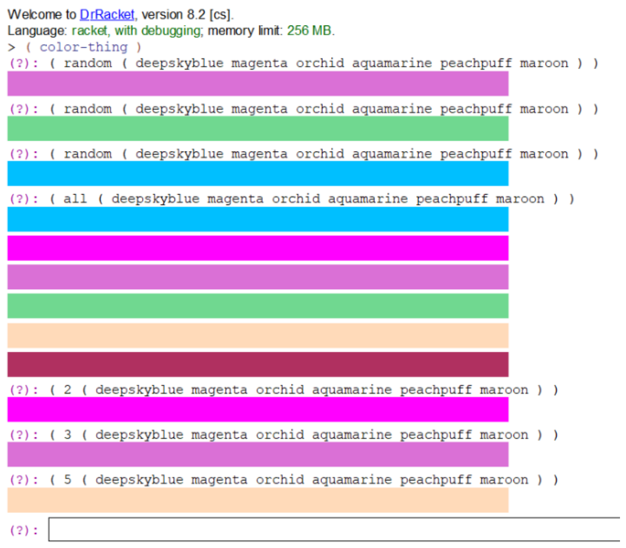
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( sampler )
(?): ( red orange yellow green blue indigo violet )
red
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
yellow
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
red
(?): ( red orange yellow green blue indigo violet )
blue
(?): ( aet ate eat eta tae tea )
tae
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
eat
(?): ( aet ate eat eta tae tea )
eta
(?): ( 0 1 2 3 4 5 6 7 8 9 )
9
(?): ( 0 1 2 3 4 5 6 7 8 9 )
3
(?): ( 0 1 2 3 4 5 6 7 8 9 )
5
(?): ( 0 1 2 3 4 5 6 7 8 9 )
9
(?): ( 0 1 2 3 4 5 6 7 8 9 )
6
(?): ( 0 1 2 3 4 5 6 7 8 9 )
0
(?): . . user break
read: illegal use of `.`
```

Problem 3b - Color Thing Interpreter

```racket
#lang racket
( require 2htdp/image )

( define ( color-thing )
   ( display "(?): " )
   ( define the-list ( read ) )
   ( define command ( list-ref the-list 0 ) )
   ( define color ( list-ref the-list 1 ) )
   ( define the-element
      ( list-ref color ( random ( length color ) ) )
      )
   ( define ( rectang color )
      ( rectangle 500 25 "solid" color )
      )
   ( define ( all color n )
      ( display ( rectang ( list-ref color n ) ) )
      ( display "\n" )
      ( cond
         ( ( < n ( - ( length color ) 1 ) )
           ( all color ( + n 1 ) )
           )
         )
      )
   ( cond
      ( ( eq? command 'random )
        ( display ( rectang the-element ) )
        ( display "\n" )
        )
      ( ( eq? command 'all )
        ( all color 0 )
        )
      ( else
        ( display ( rectang ( list-ref color ( - command 1 ) ) ) )
        ( display "\n" )
        ) )
        ( color-thing )
   )
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 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 ) )

(?):
```



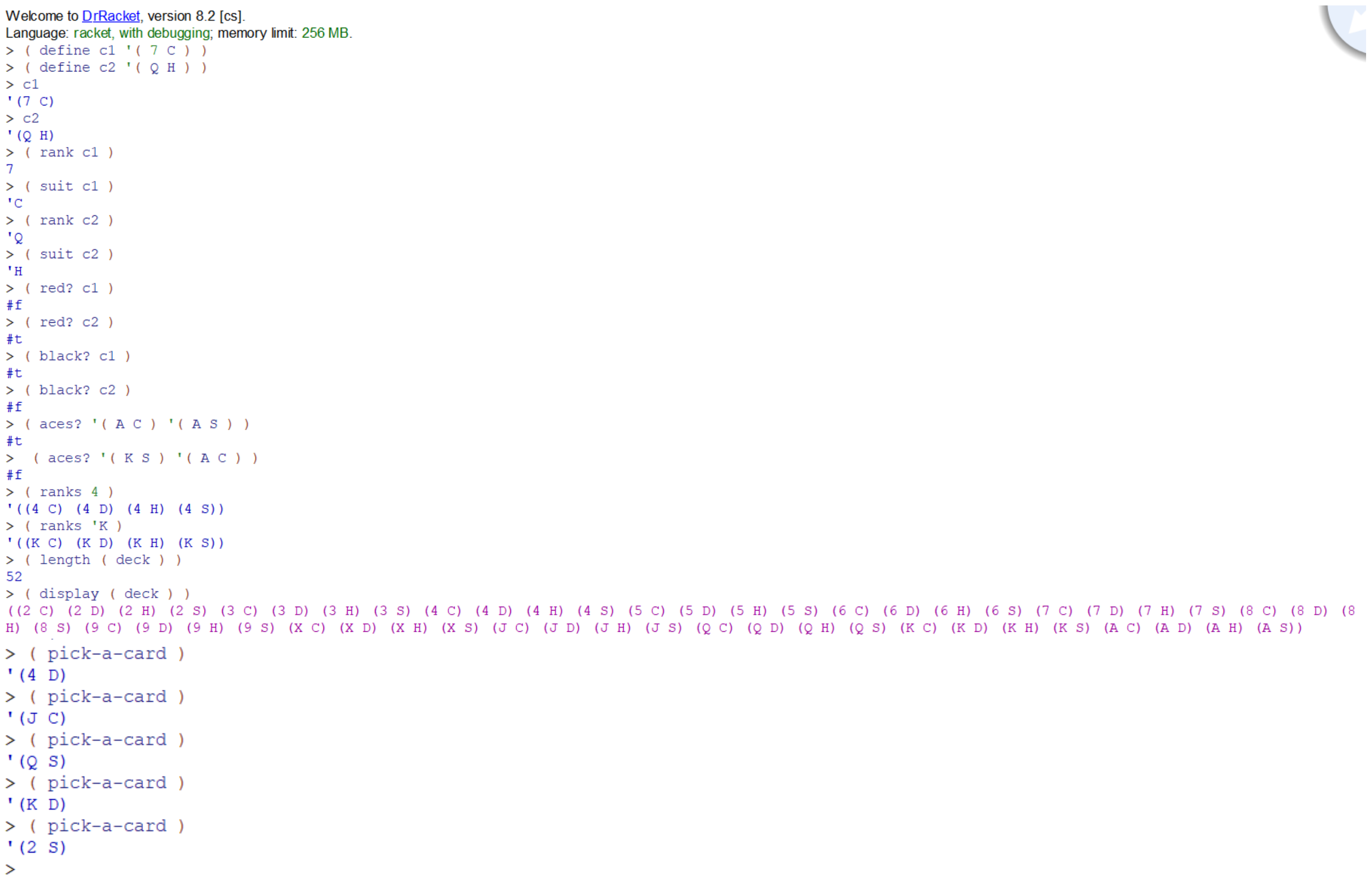
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( color-thing )
(?): ( random ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( random ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( random ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( all ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( 2 ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( 3 ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?): ( 5 ( deepskyblue magenta orchid aquamarine peachpuff maroon ) )
(?):
```

Problem 4: Two Card Poker

Problem 4a - Establishing the Card code from Lesson 6

```
#lang racket
( define ( ranks rank )
   ( list
     ( list rank 'C )
     ( list rank 'D )
     ( list rank 'H )
     ( list rank 'S )
     )
   )

( define ( deck )
   ( append
     ( ranks 2 )
     ( ranks 3 )
     ( ranks 4 )
     ( ranks 5 )
     ( ranks 6 )
     ( ranks 7 )
     ( ranks 8 )
     ( ranks 9 )
     ( ranks 'X )
     ( ranks 'J )
     ( ranks 'Q )
     ( ranks 'K )
     ( ranks 'A )
     )
   )

( define ( pick-a-card )
   ( define cards ( deck ) )
   ( list-ref cards ( random ( length cards ) ) )
   )

( define ( show card )
   ( display ( rank card ) )
   ( display ( suit card ) )
   )
```

```
( define ( rank card )
   ( car card )
   )

( define ( suit card )
   ( cadr card )
   )

( define ( red? card )
   ( or
     ( equal? ( suit card ) 'D )
     ( equal? ( suit card ) 'H )
     )
   )

( define ( black? card )
   ( not ( red? card ) )
   )

( define ( aces? card1 card2 )
   ( and
     ( equal? ( rank card1 ) 'A )
     ( equal? ( rank card2 ) 'A )
     )
   )
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( define c1 '( 7 C ) )
> ( define c2 '( Q H ) )
> c1
'(7 C)
> c2
'(Q H)
> ( rank c1 )
7
> ( suit c1 )
'C
> ( rank c2 )
'Q
> ( suit c2 )
'H
> ( red? c1 )
#f
> ( red? c2 )
#t
> ( black? c1 )
#t
> ( black? c2 )
#f
> ( aces? '( A C ) '( A S ) )
#t
>  ( 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) (X C) (X D) (X H) (X S) (J C) (J D) (J H) (J S) (Q C) (Q D) (Q H) (Q S) (K C) (K D) (K H) (K S) (A C) (A D) (A H) (A S))
> ( pick-a-card )
'(4 D)
> ( pick-a-card )
'(J C)
> ( pick-a-card )
'(Q S)
> ( pick-a-card )
'(K D)
> ( pick-a-card )
'(2 S)
>
```

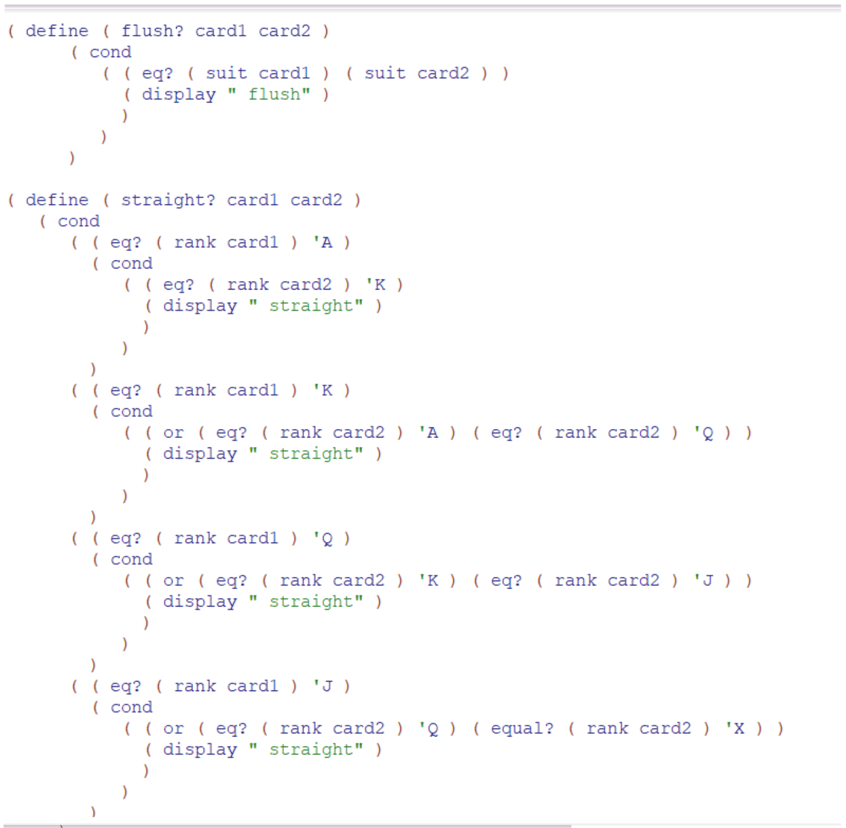
Task 4b - Two Card Poker Classifier, IR Version

```
( define ( pick-two-cards )
   ( define card1 ( pick-a-card ) )
   ( define card2 ( pick-a-card ) )
   ( cond
     ( ( eq? card1 card2 )
       ( pick-two-cards )
       )
     )
   ( list card1 card2 )
   )
```

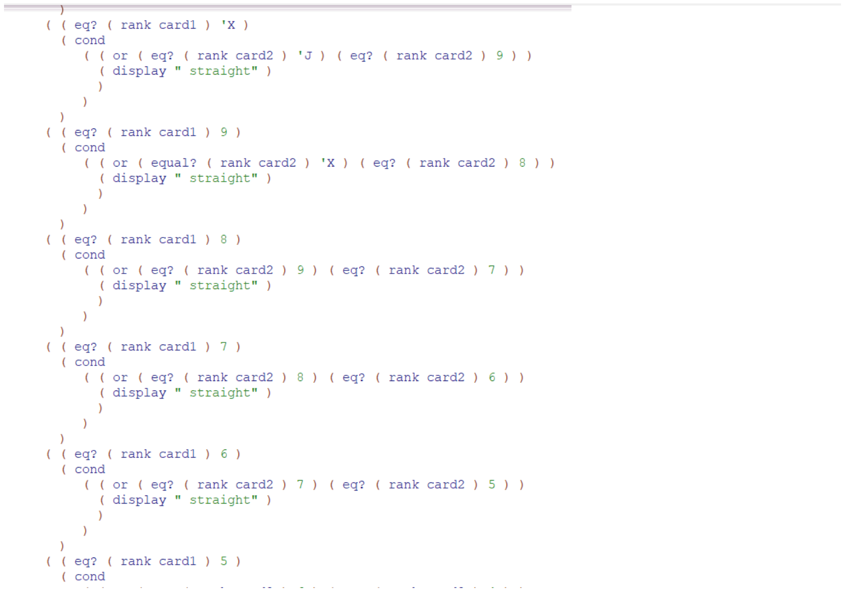
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
>  ( pick-two-cards )
'((4 S) (6 C))
>  ( pick-two-cards )
'((3 S) (3 D))
>  ( pick-two-cards )
'((8 D) (2 S))
>  ( pick-two-cards )
'((9 D) (2 C))
>  ( pick-two-cards )
'((3 D) (7 H))
>
```

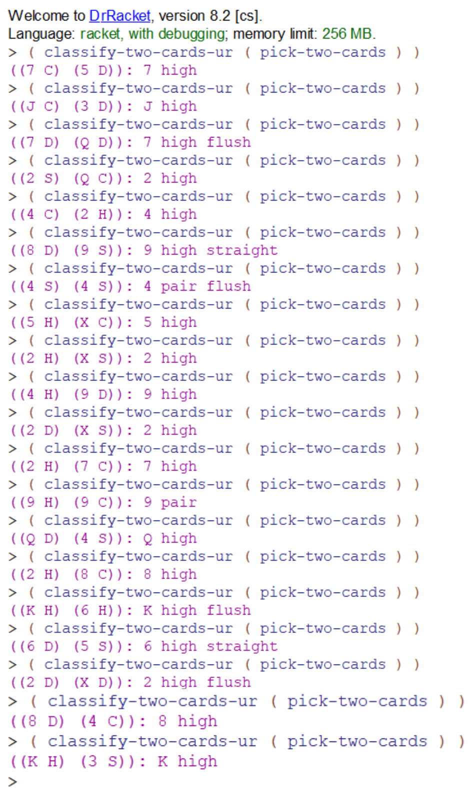
```
( define ( value rank )
  ( cond
     ( ( eq? rank 2 ) 2 )
     ( ( eq? rank 3 ) 3 )
     ( ( eq? rank 4 ) 4 )
     ( ( eq? rank 5 ) 5 )
     ( ( eq? rank 6 ) 6 )
     ( ( eq? rank 7 ) 7 )
     ( ( eq? rank 8 ) 8 )
     ( ( eq? rank 9 ) 9 )
     ( ( eq? rank 'X ) 10 )
     ( ( eq? rank 'J ) 11 )
     ( ( eq? rank 'Q ) 12 )
     ( ( eq? rank 'K ) 13 )
     ( ( eq? rank 'A ) 14 )
     )
  )

( define ( higher-rank card1 card2 )
   ( define rank1 ( value ( list-ref card1 0 ) ) )
   ( define rank2 ( value ( list-ref card2 0 ) ) )
   ( cond
      ( ( > rank1 rank2 ) ( list-ref card1 0 ) )
      ( else
        ( list-ref card2 0 )
        )
      )
   )

( trace higher-rank )
```



```
( define ( flush? card1 card2 )
      ( cond
         ( ( eq? ( suit card1 ) ( suit card2 ) )
           ( display " flush" )
           )
         )
      )

( define ( straight? card1 card2 )
   ( cond
      ( ( eq? ( rank card1 ) 'A )
        ( cond
           ( ( eq? ( rank card2 ) 'K )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 'K )
        ( cond
           ( ( or ( eq? ( rank card2 ) 'A ) ( eq? ( rank card2 ) 'Q ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 'Q )
        ( cond
           ( ( or ( eq? ( rank card2 ) 'K ) ( eq? ( rank card2 ) 'J ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 'J )
        ( cond
           ( ( or ( eq? ( rank card2 ) 'Q ) ( equal? ( rank card2 ) 'X ) )
             ( display " straight" )
             )
           )
        )
```



```
      )
      ( ( eq? ( rank card1 ) 'X )
        ( cond
           ( ( or ( eq? ( rank card2 ) 'J ) ( eq? ( rank card2 ) 9 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 9 )
        ( cond
           ( ( or ( equal? ( rank card2 ) 'X ) ( eq? ( rank card2 ) 8 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 8 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 9 ) ( eq? ( rank card2 ) 7 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 7 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 8 ) ( eq? ( rank card2 ) 6 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 6 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 7 ) ( eq? ( rank card2 ) 5 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 5 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 6 ) ( eq? ( rank card2 ) 4 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 4 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 5 ) ( eq? ( rank card2 ) 3 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 3 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 4 ) ( eq? ( rank card2 ) 2 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 2 )
        ( cond
           ( ( or ( eq? ( rank card2 ) 3 ) ( eq? ( rank card2 ) 1 ) )
             ( display " straight" )
             )
           )
        )
      ( ( eq? ( rank card1 ) 1 )
        ( cond
           ( ( eq? ( rank card2 ) 2 )
             ( display " straight" )
             )
           )
        )
      )
   )

( define ( classify-two-cards-ur card )
 ( display card ) ( display ": " )
 ( define card1 ( first card ) )
 ( define card2 ( second card ) )
   ( cond
      ( ( eq? ( rank card1 ) ( rank card2 ) )
        ( display ( rank card1 ) ) ( display " pair" )
        )
      ( ( or ( eq? ( rank card1 ) 'A ) ( equal? ( rank card2 ) 'A ) )
        ( display ( rank card1 ) ) ( display " high" )
        )
      ( ( or ( eq? ( rank card1 ) 'K ) ( equal? ( rank card2 ) 'K ) )
        ( display ( rank card1 ) ) ( display " high" )
        )
      ( ( or ( eq? ( rank card1 ) 'Q ) ( eq? ( rank card2 ) 'Q ) )
        ( display ( rank card1 ) ) ( display " high" )
        )
      ( ( or ( eq? ( rank card1 ) 'J ) ( equal? ( rank card2 ) 'J ) )
        ( display ( rank card1 ) ) ( display " high" )
        )
      ( ( or ( eq? ( rank card1 ) 'X ) ( equal? ( rank card2 ) 'X ) )
        ( display ( rank card1 ) ) ( display " high" )
        )
      ( else
        ( cond
           ( ( > ( rank card1 ) ( rank card2 ) )
             ( display ( rank card1 ) ) ( display " high" )
             )
           ( ( < ( rank card1 ) ( rank card2 ) )
             ( display ( rank card2 ) ) ( display " high" )
             )
           )
        )
      )

  ( straight? card1 card2 )
  ( flush? card1 card2 )
  )
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( classify-two-cards-ur ( pick-two-cards ) )
((7 C) (5 D)): 7 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((J C) (3 D)): J high
> ( classify-two-cards-ur ( pick-two-cards ) )
((7 D) (Q D)): 7 high flush
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 S) (Q C)): 2 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((4 C) (2 H)): 4 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 D) (9 S)): 9 high straight
> ( classify-two-cards-ur ( pick-two-cards ) )
((4 S) (4 S)): 4 pair flush
> ( classify-two-cards-ur ( pick-two-cards ) )
((5 H) (X C)): 5 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 H) (X S)): 2 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((4 H) (9 D)): 9 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 D) (X S)): 2 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 H) (7 C)): 7 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((9 H) (9 C)): 9 pair
> ( classify-two-cards-ur ( pick-two-cards ) )
((Q D) (4 S)): Q high
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 H) (8 C)): 8 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((K H) (6 H)): K high flush
> ( classify-two-cards-ur ( pick-two-cards ) )
((6 D) (5 S)): 6 high straight
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 D) (X D)): 2 high flush
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 D) (4 C)): 8 high
> ( classify-two-cards-ur ( pick-two-cards ) )
((K H) (3 S)): K high
>
```

Task 4c - Two Card Poker Classifier

```
( define ( classify-two-cards card )
 ( display card ) ( display ": " )
 ( define card1 ( first card ) )
 ( define card2 ( second card ) )
   ( cond
      ( ( and ( eq? ( rank card1 ) 'A ) ( eq? ( rank card2 ) 'A ) )
        ( display "ace pair" )
        )
      ( ( and ( eq? ( rank card1 ) 'K ) ( eq? ( rank card2 ) 'K ) )
        ( display "king pair" )
        )
      ( ( and ( eq? ( rank card1 ) 'Q ) ( eq? ( rank card2 ) 'Q ) )
        ( display "queen pair" )
        )
      ( ( and ( eq? ( rank card1 ) 'J ) ( eq? ( rank card2 ) 'J ) )
        ( display "jack pair" )
        )
      ( ( and ( eq? ( rank card1 ) 'X ) ( eq? ( rank card2 ) 'X ) )
        ( display "ten pair" )
        )
      ( ( and ( eq? ( rank card1 ) 9 ) ( eq? ( rank card2 ) 9 ) )
        ( display "nine pair" )
        )
      ( ( and ( eq? ( rank card1 ) 8 ) ( eq? ( rank card2 ) 8 ) )
        ( display "eight pair" )
        )
      ( ( and ( eq? ( rank card1 ) 7 ) ( eq? ( rank card2 ) 7 ) )
        ( display "seven pair" )
        )
      ( ( and ( eq? ( rank card1 ) 6 ) ( eq? ( rank card2 ) 9 ) )
        ( display "six pair" )
        )
      ( ( and ( eq? ( rank card1 ) 5 ) ( eq? ( rank card2 ) 5 ) )
        ( display "five pair" )
        )
      ( ( and ( eq? ( rank card1 ) 4 ) ( eq? ( rank card2 ) 4 ) )
        ( display "four pair" )


     )
   ( ( and ( eq? ( rank card1 ) 3 ) ( eq? ( rank card2 ) 3 ) )
     ( display "three pair" )
     )
   ( ( and ( eq? ( rank card1 ) 2 ) ( eq? ( rank card2 ) 2 ) )
     ( display "two pair" )
     )
   ( ( or ( eq? ( rank card1 ) 'A ) ( eq? ( rank card2 ) 'A ) )
     ( display "ace high" )
     )
   ( ( or ( eq? ( rank card1 ) 'K ) ( eq? ( rank card2 ) 'K ) )
     ( display "king high" )
     )
   ( ( or ( eq? ( rank card1 ) 'Q ) ( eq? ( rank card2 ) 'Q ) )
     ( display "queen high" )
     )
   ( ( or ( eq? ( rank card1 ) 'J ) ( eq? ( rank card2 ) 'K ) )
     ( display "jack high" )
     )
   ( ( or ( eq? ( rank card1 ) 'X ) ( eq? ( rank card2 ) 'X ) )
     ( display "ten high" )
     )
   ( ( or ( eq? ( rank card1 ) 9 ) ( eq? ( rank card2 ) 9 ) )
     ( display "nine high" )
     )
   ( ( or ( eq? ( rank card1 ) 8 ) ( eq? ( rank card2 ) 8 ) )
     ( display "eight high" )
     )
   ( ( or ( eq? ( rank card1 ) 7 ) ( eq? ( rank card2 ) 7 ) )
     ( display "seven high" )
     )
   ( ( or ( eq? ( rank card1 ) 6 ) ( eq? ( rank card2 ) 6 ) )
     ( display "six high" )
     )
   ( ( or ( eq? ( rank card1 ) 5 ) ( eq? ( rank card2 ) 5 ) )
     ( display "five high" )
     )


      ( ( or ( eq? ( rank card1 ) 4 ) ( eq? ( rank card2 ) 4 ) )
        ( display "four high" )
        )
      ( ( or ( eq? ( rank card1 ) 3 ) ( eq? ( rank card2 ) 3 ) )
        ( display "three high" )
        )
      ( ( or ( eq? ( rank card1 ) 2 ) ( eq? ( rank card2 ) 2 ) )
        ( display "two high" )
        )
      )
   ( straight? card1 card2 )
   ( flush? card1 card2 )
   )
```

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( classify-two-cards ( pick-two-cards ) )
((8 C) (8 D)): eight pair
> ( classify-two-cards ( pick-two-cards ) )
((5 D) (2 D)): five high flush
> ( classify-two-cards ( pick-two-cards ) )
((Q C) (6 H)): queen high
> ( classify-two-cards ( pick-two-cards ) )
((J C) (J H)): jack pair
> ( classify-two-cards ( pick-two-cards ) )
((5 H) (8 D)): eight high
> ( classify-two-cards ( pick-two-cards ) )
((X H) (5 D)): ten high
> ( classify-two-cards ( pick-two-cards ) )
((8 C) (Q D)): queen high
> ( classify-two-cards ( pick-two-cards ) )
((4 S) (6 C)): six high
> ( classify-two-cards ( pick-two-cards ) )
((9 C) (J S)): nine high
> ( classify-two-cards ( pick-two-cards ) )
((6 S) (3 S)): six high flush
> ( classify-two-cards ( pick-two-cards ) )
((5 C) (A H)): ace high
> ( classify-two-cards ( pick-two-cards ) )
((X H) (K S)): king high
> ( classify-two-cards ( pick-two-cards ) )
((7 D) (8 H)): eight high straight
> ( classify-two-cards ( pick-two-cards ) )
((8 S) (Q C)): queen high
> ( classify-two-cards ( pick-two-cards ) )
((6 H) (Q S)): queen high
> ( classify-two-cards ( pick-two-cards ) )
((K D) (Q C)): king high straight
> ( classify-two-cards ( pick-two-cards ) )
((4 C) (A D)): ace high
> ( classify-two-cards ( pick-two-cards ) )
((Q S) (Q H)): queen pair


> ( classify-two-cards ( pick-two-cards ) )
((X H) (K D)): king high
> ( classify-two-cards ( pick-two-cards ) )
((3 D) (3 D)): three pair flush
>
```