Racket Assignment 3: Lambda and Basic Lisp

Learning Objective:

The purpose of this lab is to teach students about the use of the built-in functions in Lisp such as the Lambda function. This assignment teaches them this by having students complete 4 tasks.

Task 1: Lambda

1a)

```
> ( ( lambda ( x ) ( cons x ( cons ( + x 1 ) ( cons ( + x 2 ) '() ) ) ) ) 5 )
'(5 6 7)
> ( ( lambda ( x ) ( cons x ( cons ( + x 1 ) ( cons ( + x 2 ) '() ) ) ) ) 0 )
'(0 1 2)
> ( ( lambda ( x ) ( cons x ( cons ( + x 1 ) ( cons ( + x 2 ) '() ) ) ) ) 108 )
'(108 109 110)
>
```

1b)

```
> ( ( lambda ( a b c ) ( cons c ( cons b ( cons a '() ) ) ) ) 'red 'yellow 'blue )
'(blue yellow red)
> ( ( lambda ( a b c ) ( cons c ( cons b ( cons a '() ) ) ) ) 10 20 30 )
'(30 20 10)
> ( ( lambda ( a b c ) ( cons c ( cons b ( cons a '() ) ) ) ) "Professor Plum" "Colonel Mustard" "Miss Scarlet" )
'("Miss Scarlet" "Colonel Mustard" "Professor Plum")
>
```

1c)

```
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
4
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
4
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
5
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
3
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
3
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
5
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
4
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
3
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
4
> ( ( lambda ( x y ) ( random x y ) ) 3 6 )
5
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
16
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
15
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
14
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
11
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
17
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
15
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
16
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
13
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
15
> ( ( lambda ( x y ) ( random x y ) ) 11 18 )
14
>
```

Task 2: List Processing Referencers and Constructors

```
> ( 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)
>
```

Task 3: Little Color Interpreter

3a)

Code:

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

Demo:

```
> ( sampler )
(?):  ( red orange yellow green blue indigo violet )
red
(?):  ( red orange yellow green blue indigo violet )
green
(?): ( red orange yellow green blue indigo violet )
orange
(?): ( red orange yellow green blue indigo violet )
blue
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
green
(?): ( red orange yellow green blue indigo violet )
green
(?): ( red orange yellow green blue indigo violet )
red
```

```
(?): ( aet ate eat eta tae tea )
eat
(?): ( aet ate eat eta tae tea )
eta
(?): ( aet ate eat eta tae tea )
tae
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
tae
(?): ( aet ate eat eta tae tea )
tea
(?): ( 0 1 2 3 4 5 6 7 8 9 )
2
(?): ( 0 1 2 3 4 5 6 7 8 9 )
3
(?): ( 0 1 2 3 4 5 6 7 8 9 )
3
(?): ( 0 1 2 3 4 5 6 7 8 9 )
9
(?): ( 0 1 2 3 4 5 6 7 8 9 )
1
(?): ( 0 1 2 3 4 5 6 7 8 9 )
1
(?): . . user break
read: illegal use of `.`
>
```

3b)

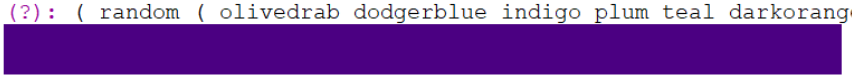
Code:

```
#lang racket
( require 2htdp/image )
( define ( color_thing )
   ( display "(?): " )
   ( define list ( read ) )
   ( define comm ( car list ) )
   ( define colors ( cadr list ) )
   ( define the-element
      ( list-ref colors ( random ( length colors ) ) )
     )

( define ( random-color )
   ( display ( rectangle 500 30 "solid" ( list-ref colors ( random ( length colors ) ) ) ) ) )
  )

( define ( select-color )
   ( display ( rectangle 500 30 "solid" ( list-ref colors ( - comm 1 ) ) ) ) )
  )

( define ( all-colors )
   ( display ( paint-colors 0 ) )
  )
```

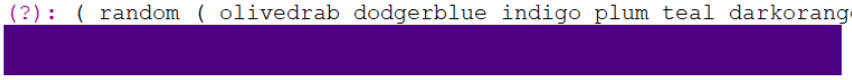
```
( define ( paint-colors n )
   ( cond
      ( ( = n ( - ( length colors ) 1 ) )
        ( display ( rectangle 500 30 "solid" ( list-ref colors n ) ) )
        ( display "\n" )
        ( color-thing )
        )
      ( else
        ( display ( rectangle 500 30 "solid" ( list-ref colors n ) ) )
        ( display "\n" )
        ( paint-colors ( + n 1 ) )
        )
      )
   )

( cond
   ( ( eq? comm 'random ) ( random-color ) )
   ( ( eq? comm 'all ) ( all-colors ) )
   ( else
     ( select-color )
     )
   )
( display "\n" )
( color_thing )
   )
```

Demo:

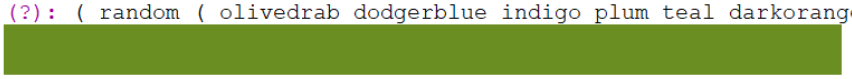
```
> ( color_thing )
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ) )
```



```
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ) )
```

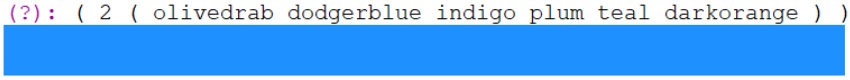


```
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ) )
```

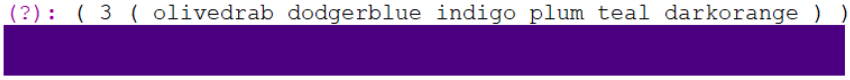


```
(?): ( all ( olivedrab dodgerblue indigo plum teal darkorange ) )
```

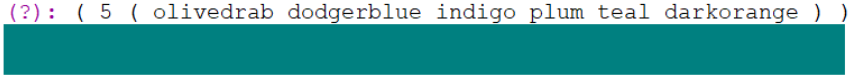
```
(?): ( 2 ( olivedrab dodgerblue indigo plum teal darkorange ) )
```



```
(?): ( 3 ( olivedrab dodgerblue indigo plum teal darkorange ) )
```

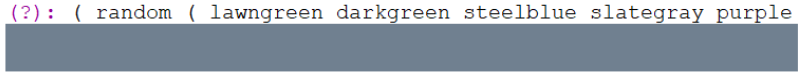


```
(?): ( 5 ( olivedrab dodgerblue indigo plum teal darkorange ) )
```

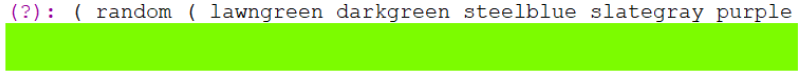


```
(?):
```

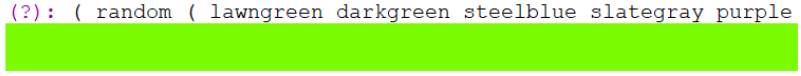
```
> ( color_thing )
(?): ( random ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



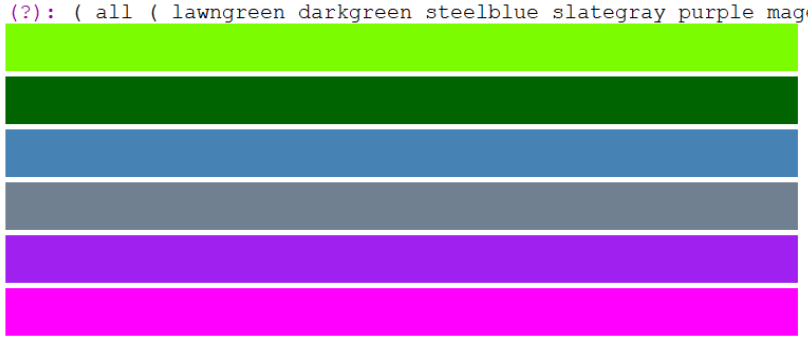
```
(?): ( random ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



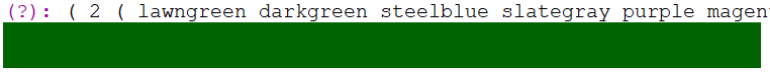
```
(?): ( random ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



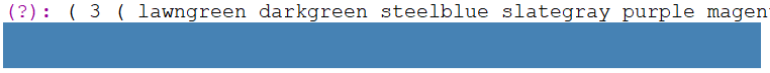
```
(?): ( all ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



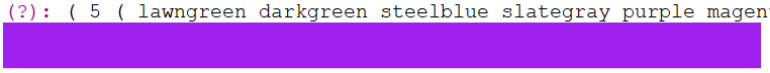
```
(?): ( 2 ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



```
(?): ( 3 ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



```
(?): ( 5 ( lawngreen darkgreen steelblue slategray purple magenta ) )
```



```
(?):
```

Task 4: Two Card Poker

4a)

Code:

```
#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 )
     )
   )
```

Demo:

```
> ( 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 H)
> ( pick-a-card )
'(8 C)
> ( pick-a-card )
'(9 H)
> ( pick-a-card )
'(9 S)
> ( pick-a-card )
'(2 C)
> ( pick-a-card )
'(6 S)
>
```

4b)

Code:

```
#lang racket
( require racket/trace )
( 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 )
    )
  )

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

( define ( higher-rank card1 card2 )
  ( display ( list card1 card2 ) )
  ( display "\n" )
  ( cond
    ( ( eq? (rank card1 ) 'A )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'A )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'K )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'K )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'Q )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'Q )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'J )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'J )
```

```
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'X )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'X )
      ( display ( rank card2 ) )
      )
    ( else
      ( cond
        ( ( eq? ( rank card1 ) ( rank card2 ) )
          ( display ( rank card1 ) )
          )
        ( ( > ( rank card1 ) ( rank card2 ) )
          ( display ( rank card2 ) )
          )
        )
      )
    )
  )

( define ( classify-two-cards-ur cards )
  ( display cards ) ( display ": " )
  ( define card1 ( first cards ) )
  ( define card2 ( second cards ) )
  ( cond
    ( ( eq? ( rank card1 ) ( rank card2 ) )
      ( display "Pair of" ) ( display ( rank card1 ) ) ( display "'s" )
      )
    ( ( eq? (rank card1 ) 'A )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'A )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'K )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'K )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'Q )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'Q )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'J )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'J )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'X )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'X )
      ( display ( rank card2 ) ) ( 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 )
  )

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

```
    ( ( eq? ( rank card2 ) '9 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '9 )
  ( cond
    ( ( eq? ( rank card2 ) 'X )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '8 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '8 )
  ( cond
    ( ( eq? ( rank card2 ) '9 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '7 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '7 )
  ( cond
    ( ( eq? ( rank card2 ) '8 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '6 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '6 )
  ( cond
    ( ( eq? ( rank card2 ) '7 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '5 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '5 )
  ( cond
    ( ( eq? ( rank card2 ) '6 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '4 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '4 )
  ( cond
    ( ( eq? ( rank card2 ) '5 )
      ( display "Straight" )
```

```
          )
        ( ( eq? ( rank card2 ) '3 )
          ( display "Straight" )
          )
        )
      )
    ( ( eq? ( rank card1 ) '3 )
      ( cond
        ( ( eq? ( rank card2 ) '4 )
          ( display "Straight" )
          )
        ( ( eq? ( rank card2 ) '2 )
          ( display "Straight" )
          )
        )
      )
    ( ( eq? ( rank card1 ) '2 )
      ( cond
        ( ( eq? ( rank card2 ) '3 )
          ( display "Straight" )
          )
        ( ( eq? ( rank card2 ) '1 )
          ( display "Straight" )
          )
        )
      )
    ( ( eq? ( rank card1 ) '1 )
      ( cond
        ( ( eq? ( rank card2 ) '2 )
          ( display "Straight" )
          )
        )
      )
    )
  )

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

Demo:

```
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 H) (J D)): JHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 S) (Q H)): QHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((7 S) (6 S)): 7HighStraightFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((3 S) (3 D)): Pair of3's
> ( classify-two-cards-ur ( pick-two-cards ) )
((Q S) (5 C)): QHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((A D) (K H)): AHighStraight
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 C) (J S)): JHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((7 H) (K H)): KHighFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 C) (9 H)): 9HighStraight
> ( classify-two-cards-ur ( pick-two-cards ) )
((J D) (X D)): JHighStraightFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((8 S) (3 S)): 8HighFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((4 C) (6 C)): 6HighFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((X C) (4 C)): XHighFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((Q S) (6 H)): QHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((6 H) (Q C)): QHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 H) (4 S)): 4High
> ( classify-two-cards-ur ( pick-two-cards ) )
((K S) (4 S)): KHighFlush
> ( classify-two-cards-ur ( pick-two-cards ) )
((9 C) (J S)): JHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((A D) (8 S)): AHigh
> ( classify-two-cards-ur ( pick-two-cards ) )
((2 D) (6 C)): 6High
>
```

4c)

Code:

```
#lang racket
( require racket/trace )
( 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 )
    )
  )

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

```
  )

( define ( higher-rank card1 card2 )
  ( display ( list card1 card2 ) )
  ( display "\n" )
  ( cond
    ( ( eq? (rank card1 ) 'A )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'A )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'K )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'K )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'Q )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'Q )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'J )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'J )
      ( display ( rank card2 ) )
      )
    ( ( eq? (rank card1 ) 'X )
      ( display ( rank card1 ) )
      )
    ( ( eq? (rank card2 ) 'X )
      ( display ( rank card2 ) )
      )
    ( else
      ( cond
        ( ( eq? ( rank card1 ) ( rank card2 ) )
          ( display ( rank card1 ) )
          )
        ( ( > ( rank card1 ) ( rank card2 ) )
          ( display ( rank card2 ) )
          )
        )
      )
    )
  )

( define ( classify-two-cards-ur cards )
  ( display cards ) ( display ": " )
  ( define card1 ( first cards ) )
  ( define card2 ( second cards ) )
  ( cond
    ( ( eq? ( rank card1 ) ( rank card2 ) )
      ( display "Pair of" ) ( display ( rank card1 ) ) ( display "'s" )
      )
    ( ( eq? (rank card1 ) 'A )
      ( display ( rank card1 ) ) ( display "High" )
```

```
      )
    ( ( eq? (rank card2 ) 'A )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'K )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'K )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'Q )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'Q )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'J )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'J )
      ( display ( rank card2 ) ) ( display "High" )
      )
    ( ( eq? (rank card1 ) 'X )
      ( display ( rank card1 ) ) ( display "High" )
      )
    ( ( eq? (rank card2 ) 'X )
      ( display ( rank card2 ) ) ( 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 )
  )

( define ( straight? card1 card2 )
  ( cond
    ( ( eq? ( rank card1 ) 'A )
      ( cond
        ( ( eq? ( rank card2 ) 'K )
          ( display "Straight" )
          )
        )
      )
    ( ( eq? ( rank card1 ) 'K )
      ( cond
        ( ( eq? ( rank card2 ) 'A )
          ( display "Straight" )
          )
        ( ( eq? ( rank card2 ) 'Q )
          ( display "Straight" )
```

```
      )
    )
  )
( ( eq? ( rank card1 ) 'Q )
  ( cond
    ( ( eq? ( rank card2 ) 'K )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) 'J )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) 'J )
  ( cond
    ( ( eq? ( rank card2 ) 'Q )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) 'X )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) 'X )
  ( cond
    ( ( eq? ( rank card2 ) 'J )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '9 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '9 )
  ( cond
    ( ( eq? ( rank card2 ) 'X )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '8 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '8 )
  ( cond
    ( ( eq? ( rank card2 ) '9 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '7 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '7 )
  ( cond
    ( ( eq? ( rank card2 ) '8 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '6 )
```

```
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '6 )
  ( cond
    ( ( eq? ( rank card2 ) '7 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '5 )
      ( display "Straight" )
      )
    )
  )
( ( eq? ( rank card1 ) '5 )
  ( cond
    ( ( eq? ( rank card2 ) '6 )
      ( display "Straight" )
      )
    ( ( eq? ( rank card2 ) '4 )
      ( display "Straight " )
      )
    )
  )
( ( eq? ( rank card1 ) '4 )
  ( cond
    ( ( eq? ( rank card2 ) '5 )
      ( display "Straight " )
      )
    ( ( eq? ( rank card2 ) '3 )
      ( display "Straight " )
      )
    )
  )
( ( eq? ( rank card1 ) '3 )
  ( cond
    ( ( eq? ( rank card2 ) '4 )
      ( display "Straight " )
      )
    ( ( eq? ( rank card2 ) '2 )
      ( display "Straight " )
      )
    )
  )
( ( eq? ( rank card1 ) '2 )
  ( cond
    ( ( eq? ( rank card2 ) '3 )
      ( display "Straight " )
      )
    ( ( eq? ( rank card2 ) '1 )
      ( display "Straight " )
      )
    )
  )
( ( eq? ( rank card1 ) '1 )
  ( cond
    ( ( eq? ( rank card2 ) '2 )
      ( display "Straight " )
      )
```

```
        )
      )
    )
  )

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

( define ( classify-two-cards cards )
  ( display cards ) ( display ": " )
  ( define card1 ( first cards ) )
  ( define card2 ( second cards ) )
  ( cond
    ( ( eq? ( rank card1 ) ( rank card2 ) )
      ( display "Pair of" ) ( display ( rank card1 ) ) ( display "'s" )
      )
    ( ( eq? (rank card1 ) 'A )
      ( display "Ace High" )
      )
    ( ( eq? (rank card2 ) 'A )
      ( display "Ace High" )
      )
    ( ( eq? (rank card1 ) 'K )
      ( display "King High" )
      )
    ( ( eq? (rank card2 ) 'K )
      ( display "King High" )
      )
    ( ( eq? (rank card1 ) 'Q )
      ( display "Queen High" )
      )
    ( ( eq? (rank card2 ) 'Q )
      ( display "Queen High" )
      )
    ( ( eq? (rank card1 ) 'J )
      ( display "Jack High" )
      )
    ( ( eq? (rank card2 ) 'J )
      ( display "Jack High" )
      )
    ( ( eq? (rank card1 ) 'X )
      ( display "Ten High" )
      )
    ( ( eq? (rank card2 ) 'X )
      ( display "Ten High" )
      )
    ( ( eq? (rank card1 ) '9 )
      ( display "Nine High" )
      )
    ( ( eq? (rank card2 ) '9 )
      ( display "Nine High" )
      )
    ( ( eq? (rank card1 ) '8 )
      ( display "Eight High" )
```

```
    )
  ( ( eq? (rank card2 ) '8 )
    ( display "Eight High" )
    )
  ( ( eq? (rank card1 ) '7 )
    ( display "Seven High" )
    )
  ( ( eq? (rank card2 ) '7 )
    ( display "Seven High" )
    )
  ( ( eq? (rank card1 ) '6 )
    ( display "Six High" )
    )
  ( ( eq? (rank card2 ) '6 )
    ( display "Six High" )
    )
  ( ( eq? (rank card1 ) '5 )
    ( display "Five High" )
    )
  ( ( eq? (rank card2 ) '5 )
    ( display "Five High" )
    )
  ( ( eq? (rank card1 ) '4 )
    ( display "Four High" )
    )
  ( ( eq? (rank card2 ) '4 )
    ( display "Four High" )
    )
  ( ( eq? (rank card1 ) '3 )
    ( display "Three High" )
    )
  ( ( eq? (rank card2 ) '3 )
    ( display "Three High" )
    )
  ( ( eq? (rank card1 ) '2 )
    ( display "Two High" )
    )
  ( ( eq? (rank card2 ) '2 )
    ( display "Two High" )
    )
  ( ( eq? (rank card1 ) '1 )
    ( display "One High" )
    )
  ( ( eq? (rank card2 ) '1 )
    ( display "One High" )
    )
  )
( straight? card1 card2 )
( straight? card1 card2 )
)
```

Demo:

```
>  ( classify-two-cards ( pick-two-cards ) )
((8 D) (J C)): Jack High
> ( classify-two-cards ( pick-two-cards ) )
((A C) (4 C)): Ace High
> ( classify-two-cards ( pick-two-cards ) )
((A S) (Q C)): Ace High
> ( classify-two-cards ( pick-two-cards ) )
((Q C) (7 D)): Queen High
> ( classify-two-cards ( pick-two-cards ) )
((A C) (J S)): Ace High
> ( classify-two-cards ( pick-two-cards ) )
((4 D) (3 S)): Four HighStraight Straight
> ( classify-two-cards ( pick-two-cards ) )
((9 H) (6 H)): Nine High
> ( classify-two-cards ( pick-two-cards ) )
((5 D) (K C)): King High
> ( classify-two-cards ( pick-two-cards ) )
((5 H) (4 C)): Five HighStraight Straight
> ( classify-two-cards ( pick-two-cards ) )
((J C) (X D)): Jack HighStraightStraight
> ( classify-two-cards ( pick-two-cards ) )
((6 S) (K D)): King High
> ( classify-two-cards ( pick-two-cards ) )
((J H) (9 C)): Jack High
> ( classify-two-cards ( pick-two-cards ) )
((3 D) (7 C)): Seven High
> ( classify-two-cards ( pick-two-cards ) )
((Q D) (K H)): King HighStraightStraight
> ( classify-two-cards ( pick-two-cards ) )
((3 D) (4 H)): Four HighStraight Straight
> ( classify-two-cards ( pick-two-cards ) )
((2 C) (2 H)): Pair of2's
> ( classify-two-cards ( pick-two-cards ) )
((4 H) (A D)): Ace High
> ( classify-two-cards ( pick-two-cards ) )
((5 H) (7 D)): Seven High
> ( classify-two-cards ( pick-two-cards ) )
((7 S) (5 H)): Seven High
> ( classify-two-cards ( pick-two-cards ) )
((K H) (X D)): King High
>
```