Task 2: Creating and Displaying a standard deck of playing cards

About this task:

The purpose of this task was simply to create cards of the four suites that comprise of a standard playing deck of cards. The deck is organized with aces at the beginning of each suit due to them being low cards in GOPS.

Demo:

[8]> (rund)

>>> Running Task 2 Demo.

>>> Testing: create-deck

--- Deck =

((ACE . CLUB) (2 . CLUB) (3 . CLUB) (4 . CLUB) (5 . CLUB) (6 . CLUB) (7 . CLUB) (8 . CLUB) (9 . CLUB) (10 . CLUB) (JACK . CLUB)

(QUEEN . CLUB) (KING . CLUB) (ACE . DIAMOND) (2 . DIAMOND) (3 . DIAMOND) (4 . DIAMOND) (5 . DIAMOND) (6 . DIAMOND) (7 . DIAMOND)

(8 . DIAMOND) (9 . DIAMOND) (10 . DIAMOND) (JACK . DIAMOND) (QUEEN . DIAMOND) (KING . DIAMOND) (ACE . SPADE) (2 . SPADE) (3 . SPADE)

(4 . SPADE) (5 . SPADE) (6 . SPADE) (7 . SPADE) (8 . SPADE) (9 . SPADE) (10 . SPADE) (JACK . SPADE) (QUEEN . SPADE) (KING . SPADE)

(ACE . HEART) (2 . HEART) (3 . HEART) (4 . HEART) (5 . HEART) (6 . HEART) (7 . HEART) (8 . HEART) (9 . HEART) (10 . HEART) (JACK . HEART)

(QUEEN . HEART) (KING . HEART))

--- Number of cards in deck = 52

NIL

[9]>

Code for the Demo:

```
( defun demo--task2 ()
  ( format t ">>> Running Task 2 Demo. ~%" )
  ( demo--create-deck )
  nil
  )

( defun demo--create-deck ()
  ( format t ">>> Testing: create-deck ~%" )
  ( setf deck ( create-deck ) )
  ( format t "--- Deck = ~A~%" deck )
  ( format t "--- Number of cards in deck = ~A~%" ( length deck
) )
  nil
  )
```

Code:

```
;-----------------------------------------------
;
; Task 2:
;
; Creating and Displaying a standard deck of cards.
;
;-----------------------------------------------

( defun create-cards ( suite &aux ranks )
```

```
  ( setf ranks '( ace 2 3 4 5 6 7 8 9 10 jack queen king ) )
  ( setf suite-duplicates ( duplicate ( length ranks ) suite ) )
  ( mapcar #'cons ranks suite-duplicates )
  )

( defun create-deck ()
  ( mapcan #'create-cards '( club diamond spade heart ) )
  )


;-------------------------------------------
;
;  Task 2 Demos
;
;

( defun demo--task2 ()
  ( format t ">>> Running Task 2 Demo. ~%" )
  ( demo--create-deck )
  nil
  )

( defun demo--create-deck ()
  ( format t ">>> Testing: create-deck ~%" )
  ( setf deck ( create-deck ) )
  ( format t "--- Deck = ~A~%" deck )
  ( format t "--- Number of cards in deck = ~A~%" ( length deck
) )
  nil
```

```
  )


;-----------------------------------------------
;
;  Task 2 Helper functions
;
;

( defun duplicate ( n lo )
  ( cond
    ( ( = n 0 )
      ()
      )
    ( t
      ( snoc lo ( duplicate ( - n 1 ) lo ) )
      )
    )
  )

( defun snoc ( o l )
  ( cond
    ( ( null l )
      ( list o )
      )
    ( t
      ( cons ( car l ) ( snoc o ( cdr l ) ) )
      )
```

)

)