Task 1: Mimic "Lisp Session: CAR, CDR and CONS"

```
CL-USER> (car '(blue red yellow) )
BLUE
CL-USER> (cdr '(blue red yellow) )
(RED YELLOW)
CL-USER> (car'(1 2) buckle my shoe))
; Evaluation aborted on #<UNBOUND-VARIABLE BUCKLE {10036956F3}>.
CL-USER>
; No value
CL-USER> (car '( ( 1 2) buckle my show ) )
(1 2)
CL-USER> ( cdr ' ( (1 2) buckle my shoe ))
(BUCKLE MY SHOE)
CL-USER> (car '("sunshine") )
"sunshine"
CL-USER> (cdr'("sunshine") )
NIL
CL-USER> (cons 'espresso '(latte cappuccino))
(ESPRESSO LATTE CAPPUCCINO)
CL-USER> (cons '(a b c) '(1 2 3) )
((A B C) 1 2 3)
CL-USER> (cons 'symbol '())
(SYMBOL)
CL-USER>
```

<u>Task 2: Mimic “Redacted Lisp Session: Three additional referencers and constructors”</u>

```
CL-USER> (setf oo-languages '(simula smalltalk java clos))
; in: SETF OO-LANGUAGES
;     (SETF OO-LANGUAGES '(SIMULA SMALLTALK JAVA CLOS))
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::OO-LANGUAGES
;
; compilation unit finished
;   Undefined variable:
;     OO-LANGUAGES
;   caught 1 WARNING condition
(SIMULA SMALLTALK JAVA CLOS)
CL-USER> oo-languages
(SIMULA SMALLTALK JAVA CLOS)
CL-USER> 'oo-languages
OO-LANGUAGES
CL-USER> (quote oo-languages)
OO-LANGUAGES
CL-USER> (car oo-languages)
SIMULA
CL-USER> (cdr oo-languages)
(SMALLTALK JAVA CLOS)
CL-USER> (car (cdr oo-languages))
SMALLTALK
CL-USER> (cadr oo-languages)
SMALLTALK
CL-USER> (cddr oo-languages)
(JAVA CLOS)
CL-USER> (first oo-languages)
SIMULA
CL-USER> (second oo-languages)
SMALLTALK
CL-USER> (third oo-languages)
JAVA
CL-USER> (nth 2 oo-languages)
```

JAVA

```
CL-USER> (setf numbers '(1 2 3))
; in: SETF NUMBERS
;     (SETF NUMBERS '(1 2 3))
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::NUMBERS
;
; compilation unit finished
;   Undefined variable:
;     NUMBERS
;   caught 1 WARNING condition
(1 2 3)
CL-USER> (setf letters '(a b c))
; in: SETF LETTERS
;     (SETF LETTERS '(A B C))
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::LETTERS
;
; compilation unit finished
;   Undefined variable:
;     LETTERS
;   caught 1 WARNING condition
(A B C)
CL-USER> (cons numbers letters)
((1 2 3) A B C)
CL-USER> (list numbers letters)
((1 2 3) (A B C))
CL-USER> (append numbers letters)
(1 2 3 A B C)
CL-USER> (list numbers (cdr numbers) (cddr numbers))
((1 2 3) (2 3) (3))
CL-USER> (append numbers (cdr numbers) (cddr numbers))
(1 2 3 2 3 3)
CL-USER> (setf elle '(ant bat cat dog eel))
; in: SETF ELLE
;     (SETF ELLE '(ANT BAT CAT DOG EEL))
;
; caught WARNING:
```

```
;   undefined variable: COMMON-LISP-USER::ELLE
;
; compilation unit finished
;   Undefined variable:
;     ELLE
;   caught 1 WARNING condition
(ANT BAT CAT DOG EEL)
CL-USER> (car (cdr (cdr (cdr elle))))
DOG
CL-USER> (nth 3 elle)
DOG
CL-USER> (setf a 'apple b 'peach c 'cherry)
; in: SETF A
;     (SETF A 'APPLE)
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::A
;
; compilation unit finished
;   Undefined variable:
;     A
;   caught 1 WARNING condition
; in: SETF A
;     (SETF B 'PEACH)
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::B
;
; compilation unit finished
;   Undefined variable:
;     B
;   caught 1 WARNING condition
; in: SETF A
;     (SETF C 'CHERRY)
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::C
;
; compilation unit finished
;   Undefined variable:
```

```
;     C
;   caught 1 WARNING condition
CHERRY
CL-USER> (cons a (cons b (cons c ())))
(APPLE PEACH CHERRY)
CL-USER> (list a b c)
(APPLE PEACH CHERRY)
CL-USER> (setf x '(red blue) y '(green yellow))
; in: SETF X
;     (SETF X '(RED BLUE))
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::X
;
; compilation unit finished
;   Undefined variable:
;     X
;   caught 1 WARNING condition
; in: SETF X
;     (SETF Y '(GREEN YELLOW))
;
; caught WARNING:
;   undefined variable: COMMON-LISP-USER::Y
;
; compilation unit finished
;   Undefined variable:
;     Y
;   caught 1 WARNING condition
(GREEN YELLOW)
CL-USER> (cons (car x ) (cons (car (cdr x)) y))
(RED BLUE GREEN YELLOW)
CL-USER> (append x y)
(RED BLUE GREEN YELLOW)
CL-USER>
```

<u>Task 3: Create a Lisp session according to specification</u>

```
CL-USER> (setf ENGLISH `(ONE TWO THREE FOUR))
(ONE TWO THREE FOUR)
 CL-USER> (setf FRENCH `(UN DEUX TROIS QUATRE))
(UN DEUX TROIS QUATRE)
CL-USER> (setf PAIR1 (list (car ENGLISH) (car FRENCH)))
(ONE UN)
CL-USER> (setf PAIR2 (list (car (cdr ENGLISH)) (car (cdr FRENCH))))
(TWO DEUX)
CL-USER> (setf PAIR3 (list (nth 2 ENGLISH) (nth 2 FRENCH)))
(THREE TROIS)
CL-USER> (setf PAIR4 (list (nth 3 ENGLISH) (nth 3 FRENCH)))
(FOUR QUATRE)
CL-USER> (setf DICTIONARY (list PAIR1 PAIR2 PAIR3 PAIR4))
((ONE UN) (TWO DEUX) (THREE TROIS) (FOUR QUATRE))
CL-USER> (setf EF-WORDS (append PAIR1 PAIR2 PAIR3 PAIR4))
(ONE UN TWO DEUX THREE TROIS FOUR QUATRE)
CL-USER> (setf ALT-WORDS (append ENGLISH FRENCH)) (ONE TWO THREE
FOUR UN DEUX TROIS QUATRE)
```