CIS 343 – Structure of Programming Languages
Winter 2021, Programming Assignment #7

Functions in LISP Language
Due Date: Wednesday, April 19, 2021

1. Define a recursive function called `my-gcd` that takes two positive integers and returns the greatest common divisor of these integers using the Euclid's algorithm. The `gcd` is defined as follows: `gcd(a, 0) = a, gcd(a, b) = gcd(b, a mod b)`. You cannot use the built-in LISP function `gcd` in your implementation.

```
(my-gcd 42 56)    returns 14
(my-gcd 12 18)    returns 6
(my-gcd 3 5)      returns 1
(my-gcd 12 60)    returns 12
(my-gcd 12 90)    returns 6
```

2. Define a function called `is-palindrome` that returns T if the list is a palindrome and NIL otherwise. A palindrome is a sequence of things that can be read the same in either direction For example,

```
(is-palindrome '(a b a))          returns T
(is-palindrome '(a b))            returns NIL
(is-palindrome '(a))              returns T
(is-palindrome '())               returns NIL
```

3. Define a function called `intlist` that takes a positive integer and returns a list of all integers between 1 and the value of the argument inclusive, in decreasing order. For example,

```
(intlist 8)  returns   (8 7 6 5 4 3 2 1)
```

   You can use recursion or iteration in your solution.

4. Define a function called `analyze` that takes a list and returns a list consisting of symbols *atom* and *list*. You can use recursion or iteration in your solution. For example,

```
(analyze '(a b c))          returns (atom atom atom)
(analyze '(a b (c d) e f))  returns (atom atom list atom atom)
(analyze '(a))              returns (atom)
(analyze '((a)))            returns (list)
```

5. Define a recursive function called `only-atoms` that takes a list as its single argument and returns T if the list contains only atoms. It should return NIL if the list contains any non-atomic values (e.g., sublists).

```
(only-atoms '(a b c))       returns T
(only-atoms '(a))           returns T
(only-atoms '(a (b) c))     returns NIL
(only-atoms '())            returns NIL
```

6. Write the same function in Question 5 using a looping (iterative) construct. You can use either LOOP, DO, DOTIMES, or DOLIST in your solution.

```
(only-atoms-iter '(a b c))       returns T
(only-atoms-iter '(a))           returns T
(only-atoms-iter '(a (b) c))     returns NIL
(only-atoms-iter '())            returns NIL
```

7. Define a function called `quad-roots` that takes three parameters *a*, *b*, and *c*, and returns a list containing the two roots of the quadratic equation `ax^2 + bx + c = 0`. For example,

```
(quad-roots 2 4 -30)   returns (3 -5) or (-5 3)
(quad-roots 1 3 -4)    returns (-4 1) or (1 -4)
(quad-roots 1 5 -6)    returns (1 -6) or (-6 1)
(quad-roots 1 2 -8)    returns (2 -4) or (-4 2)
```

8. Define a function called `rotate-from-right` that takes a list and a positive integer as arguments and returns a new list with *n* elements from right to left. You can use recursion or iteration in your solution. For example,

```
(rotate-from-right '(a b c d e) 3)    returns (C D E A B)
(rotate-from-right '(a b c d e) 2)    returns (D E A B C)
(rotate-from-right '(a b c d e) 1)    returns (E A B C D)
(rotate-from-right '(a b c d e) 4)    returns (B C D E A)
(rotate-from-right '(a b c d e) 5)    returns (A B C D E)
(rotate-from-right '(a b c d e) 7)    returns (D E A B C)
```

9. Write a recursive function called `odds` that returns every other element of a list, beginning with the first. That is, the function returns all the elements in odd-numbered positions in the list. For example,

```
(odds '(a b c d e))         returns (A C E)
(odds '(a))                 returns (A)
(odds '(a b))               returns (A)
(odds '())                  returns NIL
(odds '(a b c d e f g))     returns (A C E G)
```

10. Write the same function in Question 9 using a looping (iterative) construct. You can use either LOOP, DO, DOTIMES, or DOLIST in your solution. For example,

```
(odds-iter '(a b c d e))         returns (A C E)
(odds-iter '(a))                 returns (A)
(odds-iter '(a b))               returns (A)
(odds-iter '())                  returns NIL
(odds-iter '(a b c d e f g))     returns (A C E G)
```

Deliverables

1. Download the file **functions.lsp** from Blackboard and complete the functions defined in it.
2. Test your code on EOS machines and then upload the file **functions.lsp** on Blackboard.