COMP348/1 – Section AA – Summer 2019

ASSIGNMENT #2
Due: Sunday, June 16th by midnight 11:59 PM

Important Note

- The assignment #2 can be done with a team of 4 students. For the submission by team, ***only one copy*** of the assignment per team.
- The program file must have an extension **.lisp** called "ID_**C348SA2.lisp"** containing **only** all functions *without your own tests*; including the answer of the **exercise #1 (written in comments**).
- You must respect strictly the format of the requested functions, including the number and names of their arguments.
- Your work is to develop all the given functions with LISP language. Therefore, no new function will be accepted.
- All exercises below should be compiled, executed and return the expected results; otherwise a mark 0 (zero) will be assigned.
- The use of the "**truncate**" function *is not allowed* in this assignment. The mark zero (0) is assigned automatically by the marker.

EXERCISE #1 **(1 point)**

A) Give the results of calls of the following functions:

a. (CAR '((A (B C)) D (E F))) → ?

b. (CDR '((A (B C)) D (E F))) → ?

c. (CDR (CAR '((A (B C)) D (E F)))) → ?

d. (CAR(CDR(CDR '((A (B C)) D (E F))))) → ?

e. (CONS 'P (CONS 'O '(L))) → ?

f. (CONS (CAR '((CAR A) (CDR A))) (CAR '(((CONS A B))))) → ?

B) Use **ONLY** **CAR** and **CDR** functions to replace the "**?**" in the following statements, ***otherwise a mark of zero (0) will be assigned automatically***:

a. (? '(A B C D)) → **D**

b. (? '((A (B C)) E)) → **C**

c. (? '(((D) E) U)) → **D**

d. (? '(((D) E) U)) → **E**

EXERCISE #2 **(0.5 point)**

Write a function non-recursive called "**elementIsNumber(L)**" that tests if the second element is a number in the list. For this function, you can use the Build-in predicate function NUMBERP.

Example:

```
> (elementIsNumber  '(1  2  3  4))
T
> (elementIsNumber  '(1  a  b  4))
NIL
```

EXERCISE #3 **(0.5 point)**

Write a function non-recursive called "**elementIsList(L)**" that tests if the second element is a list in the list. For this function, you can use the Build-in predicate function CONSP.

Example:

```
> (elementIsList  '((1  2)  3  4)
NIL
> (elementIsList  '(1  a  b  4)
NIL
> (elementIsList  '( (1  (2))  3  4) )
NIL
> (elementIsList  '(1  (a)  b 4) )
T
> (elementIsList  '( (1  (2))   (3  4)) )
T
> (elementIsList  '( (1 (2))   ((3)  4)) )
T
> (elementIsList  '( (1 (2))   ((3)  (4)) ) )
T
```

EXERCISE #4 (1 point)

Write a recursive function, in LISP, called "**base8(N)**" that converts a positive integer of base 10 (decimal) to base 8 (octal) of same number.

Example:

```
> (base8  -1)          ;;; N = -1 returns a list with an element -1
( -1 )

> (base8  0)           ;;; N = 0 returns a list with an element 0
( 0 )

> (base8  7)           ;;; N = 7 returns a list with an element 7
( 7 )

> (base8  8)
(1  0 )

> (base8  15)
(1   7)

> (base8  20)
(2   4)

> (base8  204)
(3   1   4)
```

EXERCISE #5 (0.5 point)

Write a recursive function, in LISP, called "**myMember(x lst)**", i.e., **x** is member of the list **lst**.

Example:

```
> (myMember  'a  '( ) )          ;;; lst = null returns NIL
NIL
> (myMember  'b  '(a  b))
T
> (myMember  'a  '(a  b  c  d))
T
> (myMember  1  '(1  2  3  4))
T
```

EXERCISE #6 (0.5 point)

Write a recursive function, in LISP, called "**nbDigits(N)**" which takes a parameter as a positive integer number and returns the number of digits that contains that number.

Example:

```
> (nbDigits  -12)       ;;; n = -12 returns 1
1
> (nbDigits  0)         ;;; n = 0 returns 1
1
> (nbDigits  6)         ;;; n = 6 returns 1
1
> (nbDigits  9)         ;;; n = 9 returns 1
1
> (nbDigits  10)
2
> (nbDigits  1256)
4
```

EXERCISE #7 (0.5 point)

Write a recursive function, in LISP, called "**binary_length_(N)**" which takes a parameter a positive integer N and returns the number of digits that contains that integer. For this function, you can use the Build-in predicate function ZEROP.

Base case: N = 0 or N = 1 return 1

Recursive case: take the floor of the division N / 2, denoted **floor (/ N 2)** to call recursively the function.

Example:

```
> (binary_length  0)          ;;; n = 0 returns 1
1       ;;; (0)
> (binary_length 1)           ;;; n = 1 returns 1
1       ;;; (1)
> (binary_length  4)
3       ;;; (1 0 0)
> (binary_length  9)
4       ;;; (1 0 0 1)
> (binary_length  10)
4       ;;; (1 0 1 0)
```

EXERCISE #8 (1 point)

Write a recursive function, in LISP, called "**binary_list(N)**" which takes an argument a positive integer (decimal) **N** and returns a binary digits list in base 2 of that integer.

Example:

```
> (binary_list  0)                ;;; n = 0 returns an empty list → NIL
NIL
> (binary_list  -1)               ;;; n = -1 returns an empty list → NIL
NIL
> (binary_list  1)
(1)
> (binary_list  4)
(1 0 0)
> (binary_list  5)
(1 0 1)
> (binary_list  10)
(1 0 1 0)
> (binary_list  15)
(1 1 1 1)
> (binary_list  1023)
(1 1 1 1 1 1 1 1 1 1)
> (binary_list  1024)
(1 0 0 0 0 0 0 0 0 0 0)
```

EXERCISE #9 (0.5 point)

Write a recursive function, called **NTH2(n lst)**, which takes two arguments: a positive integer N and a list L and returns the N^{th} element of the list.

Example:

```
> (nth2  1  '( ) )                ;;; if lst = null  returns NIL
NIL
> (nth2  1  '(12   4   65   3) )  ;;; if n = 1  returns the first element of the list
12
> (nth2  2  '(12   4   65   3) )  ;;; if n = 2  returns the second element of the list
4
> (nth2  3  '(12   4   65   3) )  ;;; if n = 3  returns the third element of the list
65
> (nth2  4  '(12   4   65   3) )  ;;; if n = 4  returns the fourth element of the list
3
```

EXERCISE #10 (0.5 point)

Write a recursive function, called **NTHCDR2(n lst)**, which takes two arguments: a positive integer N and a list L and returns the N^{th} CDR of the list.

Example:

```
> (nthcdr2  0  '( ) )                ;;; lst = null  returns NIL
NIL
> (nthcdr2  0  '(12   4   65   3) )  ;;; n = 0  returns lst
(12   4   65   3)
> (nthcdr2  1  '(12   4   65   3) )
(4   65   3)
> (nthcdr2 2  '(12   4   65   3) )
(65   3)
> (nthcdr2  3  '(12   4   65   3) )
(3)
> (nthcdr2  4  '(12   4   65   3) )
NIL
```

EXERCISE #11 (0.5 point)

Write a recursive function, called **NTHCAR2(n lst)**, which takes two arguments: a positive integer N and a list L and returns the N^{th} CAR of the list.

Example:

```
> (nthcar2  0  '( ) )                ;;; lst = null returns NIL
NIL
> (nthcar2  0  '(12   4   65   3) )  ;;; n = 0 returns NIL
NIL
> (nthcar2  1  '(12   4   65   3) )
(12)
> (nthcar2  2  '(12   4   65   3) )
(12   4)
> (nthcar2  3  '(12   4   65   3) )
(12   4   65)
> (nthcar2  4  '(12   4   65   3) )
(12   4   65   3)
```

Evaluation Criteria or Assignment #2 (7.5 points)

Tasks (7.5 points)	
Exercise #1 : execution of functions CAR and CDR	1 pt.
Exercise #2 : elementIsNumber(L)	0.5 pt.
Exercise #3 : elementIsList(L)	0.5 pt.
Exercise #4 : base8(N)	1 pt.
Exercise #5 : myMember(x lst)	0.5 pt.
Exercise #6 : nbDigits(N)	0.5 pt.
Exercise #7 : binary_length_(N)	0.5 pt.
Exercise #8 : binary_list(N)	1 pt.
Exercise #9 : NTH2(n lst)	0.5 pt.
Exercise #10 : NTHCDR2(n lst)	0.5 pt.
Exercise #11 : NTHCAR2(n lst)	0.5 pt.
General correctness of code	0.5 pt.