COMP-301 PS-2 **(Name-Surname: Barış Kaplan, KU ID Number: 69054)**

PROBLEM-1 :

The result of (list 1 (list 2 3) 4) : '(1 (2 3) 4)

The result of (list (list (list 1 2) 3 4) (+ 2 3) (- 8 2)) : ' (((1 2) 3 4) 5 6)

The result of (list (list '())) : ' ((( )))

The result of (cons 1 2) : ' (1 . 2)

The result of (cons 1 '(2)) : ' (1 2)

The result of (car (cons 1 '(2))) : 1

The result of (cdr (cons 1 '(2))) : ' (2)

The result of (car (cdr (cdr ' (1 2 3 4)))) : 3

The result of (car (car (cdr (cdr '(a b (c d e)))))) : ' c

The result of (cons 1 (cons 'a 'b)) : ' (1 a . b)

The result of (cdr (cons 'a '(b c))) : ' (b c)

PROBLEM-2 :

' (1 (2 . 3) 4) : It is the result of (list 1 (cons 2 3) 4)

' (1 2) : It is the result of (cons 1 '(2))

' (1 . 2) : It is the result of (cons 1 2)

' (1 2 (3 4) (5 6)) : It is the result of (list 1 2 (list 3 4) (list 5 6))

' (() 1 2) : It is the result of (list '() 1 2)

' (1 2 . 3) : It is the result of (cons 1 (cons '2 '3))

'(a b . c) : It is the result of (cons 'a (cons 'b 'c))

'(a b (c d) (e . f)) : It is the result of (list 'a 'b (list 'c 'd) (cons 'e 'f))

Problem-3:

Part-A:

```
(define (even-numbers my_custom_lst)
  (if (null? my_custom_lst)
      null
  (if(list? (car my_custom_lst))
     (cons (even-numbers (car my_custom_lst)) (even-numbers (cdr my_custom_lst)))
     (if(= (custom_remainder_imp (car my_custom_lst) 2) 1)
     (even-numbers (cdr my_custom_lst))
     (cons (car my_custom_lst) (even-numbers (cdr my_custom_lst)))))))


(define custom_remainder_imp (lambda (initialNum secondNum)
  (- initialNum (* (floor (/ initialNum secondNum)) secondNum))))
```

Part-B:

```
(define (substitute ch1 ch2 custom_word)
  (if(and (null? ch1) (null? ch2))
     custom_word
  (if(null? custom_word)
     null
  (if(list? (car custom_word))
     (cons (substitute ch1 ch2 (car custom_word)) (substitute ch1 ch2 (cdr custom_word)))
  (if(equal? ch2 (car custom_word))
     (cons ch1 (substitute ch1 ch2 (cdr custom_word)))
     (cons (car custom_word) (substitute ch1 ch2 (cdr custom_word))))))))
```

Problem-4:

```
(define incrementby (lambda (n) (lambda (x) (+ x n))))
(incrementby 2)
((lambda (n) (lambda (x) (+ x n))) 2)
 (lambda (x) (+ x 2))
(define f1 (incrementby 6))
(f1 4)
```

The result of the above code block is: 10

```
(define f2 (lambda (x) (incrementby 6)))
(f2 4)
((f2 4) 6)
```

The result of the above code block is: 12

```
(define (compose f g) (lambda (x) (f (g x))))
((compose (lambda (p) (if p "hi" "bye"))
(lambda (x) (> x 0))) -5)
```

The result of the above code block is: “bye”

```
(define add2 (lambda (n) (+ n 2)))
(define add4 (compose add2 add2))
(add4 7)
```

The result of the above code block is: 11

Problem-5:

```
(define double (lambda(opr)
   (lambda (my_num) (opr (opr my_num)))))

(define inc (lambda(my_num) (+ 1 my_num)))
```

21 is returned by the following expression: (((double (double double)) inc) 5)

Little explanation (2= 2^1 , 4= 2^2 , 16= 2^4):

((double inc) 5) ; It returns 7. 5+2= 7. It applied the procedure called inc to 5 2 times.

(((double double) inc) 5) ; It returns 9. 5+ 2^2= 5+4= 9 . It applied the procedure called inc to 5 4 times.

(((double (double double)) inc) 5) ; It returns 21. 5+ 2^4= 5+ 16= 21. It applied the procedure called inc to 5 16 times.