Problem Set 2
COMP301 Fall 2021
Week 3: 11.10.2021 - 15.10.2021

Instructions:

• Submit your answers to the Blackboard PS2 assignment until October 15th Friday, at 23.59.
• Please submit only **one single PDF file**, where all your codes for each of the parts are included.
• Name your submission file as id_username_ps2.pdf
  (Example: 00000_akutuk21_ps2.pdf).

Problem 1: In this problem, you will practice the concepts of list, cons, car and cdr. Given the code block below, please find the results printed on the interpreter in response to each expression:

```
(list 1 (list 2 3) 4)
(list (list (list 1 2) 3 4) (+ 2 3) (- 8 2))
(list (list '()))
(cons 1 2)
(cons 1 '(2))
(car (cons 1 '(2)))
(cdr (cons 1 '(2)))
(car (cdr (cdr `(1 2 3 4))))
(car (car (cdr (cdr '(a b (c d e))))))
(cons 1 (cons 'a 'b))
(cdr (cons 'a '(b c)))
```

Problem 2: In this problem, you will practice the concepts of list and cons. This time you are given with the results of the interpreter, and you are supposed to write the expression which leads to that result to be printed.

```
'(1 (2 . 3) 4)
'(1 2)
'(1 . 2)
'(1 2 (3 4) (5 6))
'(() 1 2)
'(1 2 . 3)
'(a b (c d) (e . f))
'(a b . c)
```

Problem 3: In this section, you will focus on procedures and recursion to practice car and cdr.

Part A. Write a recursive procedure called *even-numbers*, which removes all odd numbers in the given list and returns the same list with **even numbers only**. You can test your implementation with the following code piece:

```
(even-numbers ’(1 0 3 4)) ; returns '(0 4)
(even-numbers ’(1 1 2 3 4 7 4)) ; returns '(2 4 4)
```

```
(even-numbers '()) ; returns '()
```

Part B. Write a procedure *substitute* that takes three arguments, two words and a sentence. It should return a version of the sentence, but with every instance of the second word replaced with the first word. You can test your implementation with the following code piece:

```
(substitute 'o 'a '(c a m p 3 a 1))
 ; returns (c o m p 3 o 1)

(substitute 'a 'b '())  ; returns '()
```

Problem 4: In this problem, you will practice the higher order procedures. Given the code block below, please find the results printed on the interpreter in response to each expression:

```
(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)

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

(define (compose f g) (lambda (x) (f (g x))))
((compose (lambda (p) (if p "hi" "bye"))
            (lambda (x) (> x 0))) -5)
(define add2 (lambda (n) (+ n 2)))
(define add4 (compose add2 add2))
(add4 7)
```

Problem 5: In this section, you will write a higher order procedure. You may refer back to the Problem 3 to have an idea.

Part A. Write a procedure *double* that takes a procedure of one argument as argument and returns a procedure that applies the original procedure twice. For example, if inc is a procedure that adds 1 to its argument, then (double inc) should be a procedure that adds 2.

Also, write that what value is returned by the following expression?

```
(((double (double double)) inc) 5)  ; returns ?
```

(Note that if you want to run your *double* procedure, it may helpful if you implement *inc* procedure as well.)