Problem Set 3
COMP301 FALL 2022
Week 3: 17.10.2022 - 21.10.2022

Instructions:

- Submit your answers to the Blackboard PS3 assignment until October 22 Saturday, at 23.59.
- Please submit only **one single PDF file**, where all of your codes for each of the parts are included.
- Name your submission file as *id_username_ps3.pdf*
  (Example: *00000_yhizir19_ps3.pdf*).

Review Part.

Problem 1: Define the set of square numbers (0, 1, 4, 9, 16 ...) by using top-down, bottom-up and rules of inference approach. You can check if an element of the set can be decomposed into an equation which satisfies the properties. For equations, you may need to use an additional variable and demonstrate the relation between the variables. In your answer do not use the square operator. Hint: $(x + y)^2 = x^2 + 2xy + y^2$

Coding Part.

Problem 2: Given a list and a number n, implement a procedure named "repeatN" that repeats each element of the list n times and return the new list. (Hint: you can use append instead of cons to achive the results below)

```
(repeatN '(1 2 3) 2) ; returns (1 1 2 2 3 3)
(repeatN '(1 2) 4) ; returns (1 1 1 1 2 2 2 2)
```

Problem 3: Given a list, write a procedure called (sum-odd list) by just using the higher order procedures map-filter-reduce such that the procedure returns the sum of all odd numbers in a list.

```
(sum-odd '(1 2 3)) ; returns 4
(sum-odd '(1 2 3 4 5)) ; returns 9
```

Problem 4: (Challenging). In this part you will implement your own higher order procedure that will combine filter and reduce. First implement a procedure named "isIn?" that takes a list and a number n which will return true if the number is in the list, false otherwise. (If you want a challenge you can try to take nested list into account as well. (ex: (isIn?-nested '(1 2 (3 4) 5) 3) –> #t)). Secondly, implement the higher order procedure named "myProc", (myProc pred lst1 lst2 op init) such that the predicate (pred) needs to accept a list and an element and returns a boolean. The procedure "myProc" returns the computed single result that is reduced while operation on the lst1 with op. (**Hint**: while using the pred give lst2 as it is and iterate over lst1)

```
(myProc isIn? '(1 5 8 13) '(1 2 3 4 5) + 0) ;returns 6 (1,5 isIn the lst2 1+5=6)
(myProc isIn? '(1 5 8 13) '(1 5 10 12) * 1) ; returns 5
(myProc (lambda (ls2 n) (not (isIn? ls2 n))) '(1 5 8) '(1 2 5) + 0) ; returns 8
(myProc isIn?-nested '(1 5 8 13) '(1 (2 (3 4) 8)) / 1) ; returns 1/8
```