# Problem Set 2
# COMP301 FALL 2022
## Week 2: 10.10.2022 - 14.10.2022

**Instructions:**

- Submit your answers to the Blackboard PS2 assignment until October 15th 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_ps2.pdf* (Example: *00000_shamdan17_ps2.pdf*).

**Question Answering Part.** In this section, you will answer the question in a paragraph.

**Problem 1:** What is the difference between recursive and iterative procedures? Explain their definitions and advantages over each other.

**Coding Part.** In this section, you will focus on car, cdr, cons and recursive, iterative procedures.

**Problem 2a:** Given a list, implement a *recursive* procedure named "even-odd" that takes the given list in which you need to add when you see an even number and subtract when you see an odd number. Then, the procedure returns the final value which is the calculation of the additions of evens and subtractions of odds. You can use modulo procedure to check whether the number is even or odd. You can test your implementation with the following code piece:

```
(even-odd '(1 2 3 4)) ; returns 2
(even-odd '(1 2 3 5 8 13)) ; returns -12
```

**Problem 2b:** Implement a *iterative* version of the "even-odd" procedure called "even-odd-iter". You can test your implementation with the following code piece:

```
(even-odd-iter '(1 2 3 4)) ; returns 2
(even-odd-iter '(1 2 3 5 8 13)) ; returns -12
```

**Problem 3: (Challenging).** Given a list, implement a procedure named "swap" that takes the given list and two integers $a$ and $b$, which are the indices of the given list to be swapped. Then, the procedure returns the swapped version of the given list. Check the *car* and *cdr* operations to see how to take subsets of the given list. Another very useful operation is ***append***, which can be used to join two lists. To make it simpler, we do not test the edge cases such as giving empty list or giving indices which are out of range. Also, you can assume that integer $a$ is always given smaller than integer $b$. You can test your implementation with the following code piece:

```
(swap 1 2 '(1 2 3 4)) ; returns '(1 3 2 4)
(swap 0 2 '(1 2 3 5 8 13)) ; returns '(3 2 1 5 8 13)
```