**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 2

**З дисципліни:** *“Декларативне програмування”*

**На тему:** *“Розроблення найпростіших програм функційною мовою”*

**Лектор:**

доцент каф. ПЗ

Левус Є.В.

**Виконав:**

ст. гр. ПЗ-45

Хруставчук М.Л.

**Прийняв:**

доцент каф. ПЗ

ТушницькийР.Б.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2025 р.

∑= \_\_\_\_\_ .

Львів – 2025

**Тема роботи:** Розроблення найпростіших програм функційною мовою.

**Мета роботи:** Вивчити використовувати базові примітиви функційної мови для створення найпростіших програм опрацювання S-виразів.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Функційне (декларативне) програмування – це парадигма, у якій програма розглядається як набір функцій, що описують що потрібно обчислити, а не як це зробити. Основна ідея – опис залежностей між вхідними даними та результатом без зміни стану системи. Ключові принципи: чисті функції, рекурсія, композиція функцій, відсутність побічних ефектів. У таких програмах змінні не змінюють своїх значень, а всі обчислення виконуються через передачу аргументів.

Однією з перших функційних мов є Lisp, створена Джоном Маккарті. Сучасні її діалекти – Scheme та Racket, які широко застосовуються у навчанні та наукових дослідженнях. У мові Racket основними структурами даних є атом, пара і список, що дозволяє представляти ієрархічні структури. Для роботи зі списками використовуються базові функції-примітиви: CAR (вибір голови списку), CDR (вибір хвоста), CONS (створення нового списку), QUOTE (блокування обчислень), EQ? (порівняння), PAIR? (перевірка типу).

Функції в Racket визначаються через ключові слова define або lambda, що дозволяє будувати складні вирази за принципом композиції. Цикли у функційному програмуванні замінюються рекурсією, яка викликає функцію саму себе до досягнення базового випадку. Такий підхід забезпечує логічну чистоту, формальність і математичну точність опису алгоритмів, що є основою декларативного стилю програмування.

**ЗАВДАННЯ**

1. Написати функційну програму, що вирішує поставлене завдання, без використання повторюваних обчислень.
2. Варіант 13. У вхідному списку з чотирьох елементів кожен атом є однією з останніх трьох літер англійського алфавіту. Якщо третій та четвертий елементи списку однакові, повертати значення – список з інших елементів цього списку. Інакше – пару з інших елементів цього списку. Список іншої довжини, не з заданих літер, атом, пару вважати неправильними вхідними даними.

**КОД ПРОГРАМИ**

Файл Lab\_02.rkt

(define (IsProperList? lst)

(cond

((eq? (pair? lst) #f) #f)

((eq? (cdr lst) '()) #t)

((pair? (cdr lst)) (IsProperList? (cdr lst)))

(#t #f)

)

)

(define (IsValidLength? lst)

(cond

((eq? lst '()) #f)

((eq? (cdr lst) '()) #f)

((eq? (cddr lst) '()) #f)

((eq? (cdddr lst) '()) #f)

((eq? (cddddr lst) '()) #t)

(#t #f)

)

)

(define (IsValidLetters? lst)

(cond

((eq? lst '()) #t)

((eq? (car lst) 'x) (IsValidLetters? (cdr lst)))

((eq? (car lst) 'y) (IsValidLetters? (cdr lst)))

((eq? (car lst) 'z) (IsValidLetters? (cdr lst)))

(#t #f))

)

(define (IsValidList? lst)

(cond

((eq? (IsProperList? lst) #f) #f)

((eq? (IsValidLength? lst) #f) #f)

((eq? (IsValidLetters? lst) #f) #f)

(#t #t))

)

(define (ModifyList lst)

(cond

((eq? (IsValidList? lst) #t)

(cond

((eq? (caddr lst) (cadddr lst))

(cons (car lst) (cons (cadr lst) '())))

(#t

(cons (car lst) (cadr lst)))))

(#t 'Not-Valid-List)))

(ModifyList '(z y x x))

(ModifyList '(x y z x))

(ModifyList '(x y z a))

(ModifyList '(x y z x 2))

(ModifyList '(x . y))

(ModifyList 'x)

**Результати роботи програми**

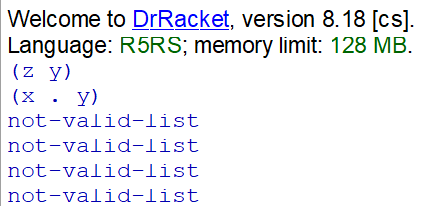


Рис. 1. Результат модифікації списку відповідно до індивідуального завдання

**ВИСНОВКИ**

У ході виконання роботи я ознайомився з принципами функційного програмування та мовою Racket, яка реалізує декларативний підхід. Було розглянуто базові функції-примітиви (CAR, CDR, CONS, QUOTE, EQ?, PAIR?) та принцип побудови програм через рекурсію і композицію функцій. Практична частина дозволила закріпити розуміння роботи зі списками та структурою S-виразів. Отримані знання поглибили розуміння різниці між імперативним і декларативним стилем програмування.