Міністерство освіти та науки України

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

**ЗВІТ**

про виконання

Модульної Контрольної Роботи №1

З дисципліни:

«**Мультипарадигменне програмування**»

**Варіант 1**

**Виконав:** студент групи ІП-51

Булатов Дмитро Эгорович

**Перевірив**: викл.

Баклан Ігор Всеволодович

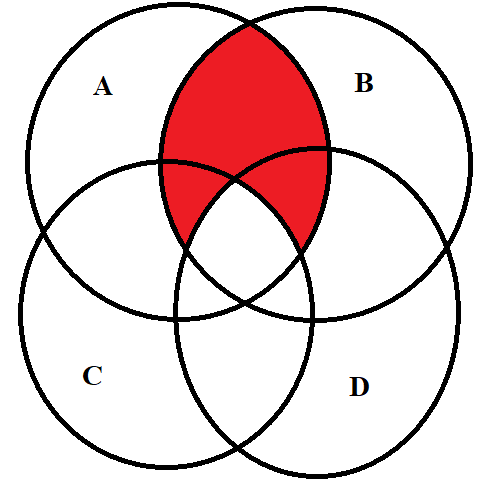
Київ 2017

**Завдання**: На Лісп написати функцію (**ОДНУ**) , яка реалізує суперпозицію теоретико-множинних операцій у відповідності до варіанта з таблиці.

А, B, C, D - списки. В функції передбачити перетворення списків (можливо й складної структури) до вигляду множин з подальшим виконанням операцій над ними.

Для написання функції використовувати виключно функції з базового набору (CDR, CAR, COND, EQUAL, LIST, CONS, LISTP, NULL). Не використовуйте функції IF, INTERSECTION, DIRRERENCE, UNION та інші, що не є базовими.

**Діаграмма**:



**Код програми:**

(defun var1(A B C D)

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;difference

(cond

((null A) '())

((null B) A)

((atom A)

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

A

B

)

nil

)

(T A)

)

)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

B

)

(funcall F (cdr A) B F)

)

(T (append (list (car A)) (funcall F (cdr A) B F)))

)

)

)

)

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;intersect

(cond

((null A) '())

((null B) '())

((atom A)

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

A

B

)

A

)

(T nil)

)

)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

B

)

(append (list (car A)) (funcall F (cdr A) B F))

)

(T (funcall F (cdr A) B F))

)

)

)

)

(funcall

(lambda (F A)

(funcall F A F)

)

(lambda (A F) ;simplify

(cond

((null A) '())

((atom A) A)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

(cdr A)

)

(funcall F (cdr A) F)

)

(T (append (list (car A)) (funcall F (cdr A) F)))

)

)

)

)

A

)

(funcall

(lambda (F A)

(funcall F A F)

)

(lambda (A F) ;simplify

(cond

((null A) '())

((atom A) A)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

(cdr A)

)

(funcall F (cdr A) F)

)

(T (append (list (car A)) (funcall F (cdr A) F)))

)

)

)

)

B

)

)

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;intersect

(cond

((null A) '())

((null B) '())

((atom A)

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

A

B

)

A

)

(T nil)

)

)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

B

)

(append (list (car A)) (funcall F (cdr A) B F))

)

(T (funcall F (cdr A) B F))

)

)

)

)

(funcall

(lambda (F A)

(funcall F A F)

)

(lambda (A F) ;simplify

(cond

((null A) '())

((atom A) A)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

(cdr A)

)

(funcall F (cdr A) F)

)

(T (append (list (car A)) (funcall F (cdr A) F)))

)

)

)

)

C

)

(funcall

(lambda (F A)

(funcall F A F)

)

(lambda (A F) ;simplify

(cond

((null A) '())

((atom A) A)

(T

(cond

(

(funcall

(lambda (F A B)

(funcall F A B F)

)

(lambda (A B F) ;member

(cond

((null B) nil)

(T

(cond

((equal A (car B)) T)

(T (funcall F A (cdr B) F))

)

)

)

)

(car A)

(cdr A)

)

(funcall F (cdr A) F)

)

(T (append (list (car A)) (funcall F (cdr A) F)))

)

)

)

)

D

)

)

)

)

(print

(var1

'(Q E Y S D G Q Y)

'(R E Y S K G)

'(J Y D P L)

'(D P Y)

))

(print

(var1

'()

'()

'()

'()

))

(print

(var1

'(H E L L O)

'(W O R L D)

'(L I S P)

'(M K R 1)

))

(print

(var1

'(H E L L O)

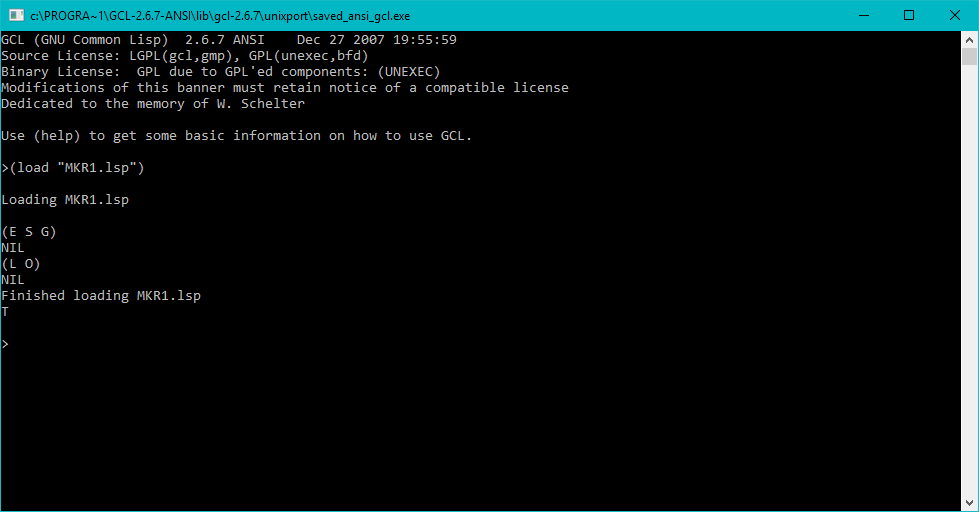
'(W O R L D)

'(H E L L O)

'(W O R L D)

))

**Скріншот програми:**



**Висновок:**

В цій модульнії контрольній я написав функцію яка реалізує суперпозицію теоретико-множинних операцій у відповідності до варіанта з таблиці. Для написання функції я використовував виключно функції з базового набору (CDR, CAR, COND, EQUAL, LIST, CONS, LISTP, NULL) та не використовував функції IF, INTERSECTION, DIRRERENCE, UNION та інші, що не є базовими.