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

Лабораторної роботи №2

З дисципліни:

«**Мультипарадигменне програмування**»

**Тема: «Опис та виклик функцій в мові Lisp»**

**Варіант 1**

**Виконав:** студент групи ІП-51

Булатов Дмитро Эгорович

**Перевірив**: викл.

Баклан Ігор Всеволодович

Київ 2017

**Мета роботи:**

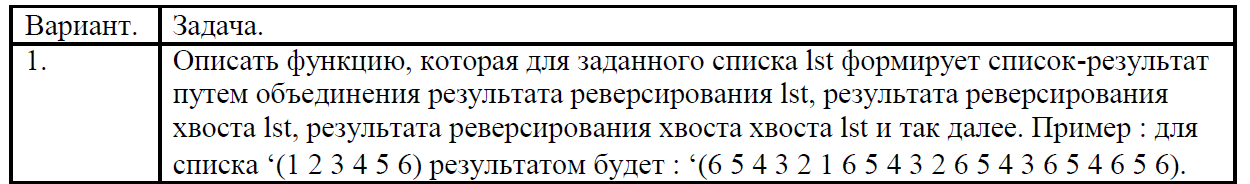
Целью работы является изучение основных правил написания рекурсивных функций в функциональном языке и изучение основных методов разработки функциональных программ с позиций Строго Функционального Языка.

**Основні задачі:**

* На примере GNU Common Lisp‟а (GCLisp'а) научиться формулировать условие завершения рекурсии, описывать формирование результата функции и новых значений аргументов для рекурсивного вызова;
* Получить практические навыки работы со списочными структурами в выбранной реализации языка Лисп;
* Освоить приемы нисходящего и восходящего проектирования функциональных программ;
* Научиться выделять основные и вспомогательные функции с учетом разбиения задачи на подзадачи;
* Овладеть приемами использования накапливающих параметров во вспомогательных функциях;
* Ознакомиться с упреждающим использованием результата вызова функции.

**Завдання 1**

Описать функцию в соответствии со своим вариантом задания из Таблицы 1, вариант выдает преподаватель.



**Код програми:**

(defun rev(A)

(cond

((atom A) A)

(T (append (rev (cdr A)) (list (car A))))

)

)

(defun T1(A)

(cond

((atom A) A)

(T (append (rev A) (T1 (cdr A))))

)

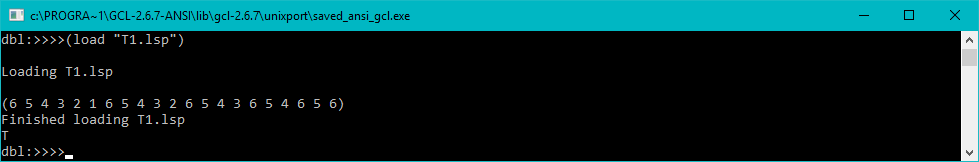
)

(print

(T1 '(1 2 3 4 5 6))

)

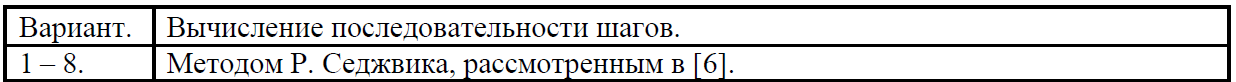
**Скріншот програми:**



В цій програмі я створив іменовані функції що перевертає список та рекурсивно додає до кінця списку поточний список

**Завдання 2**

Написать программу сортировки списка методом Шелла. Вычисление последовательности шагов сортировки производится в соответствии с вариантом в Таблице 2.



**Код програми:**

(defun insert (A B)

(cond ((null A) (LIST B))

((> (CAR A) B) (CONS B A))

(T (CONS (CAR A) (insert (cdr A) B)))))

(defun insertionsort(A B)

(cond

((null A) B)

(T (insertionsort(cdr A) (insert B (CAR A))))

)

)

(defun shellround (A B C)

(cond

((null A) C)

(T

(cond

((> B (LENGTH A)) (APPEND C (insertionsort A nil)))

(T (APPEND C (insertionsort(subseq A 0 B) nil) (shellround (subseq A B (LENGTH A)) B C)))

)

)

)

)

(defun shellsort (A B)

(cond

((null (cdr B)) (shellround A (CAR B) '()))

(T (shellsort (shellround A (CAR B) '()) (cdr B)))

)

)

(defun sendgewick()

'(1 8 23 77 281 1073 4193 16577 65921 262913)

)

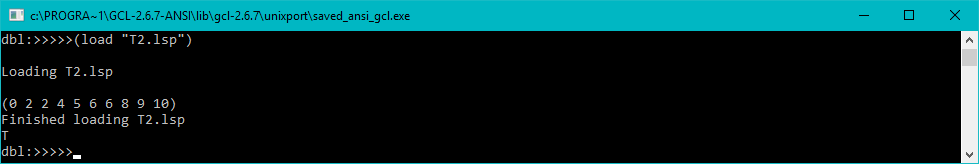
(defun sortWithShellAndSedgewick (A)

(shellsort A (sendgewick))

)

(print (sortWithShellAndSedgewick '(5 6 8 4 2 0 2 6 10 9)))

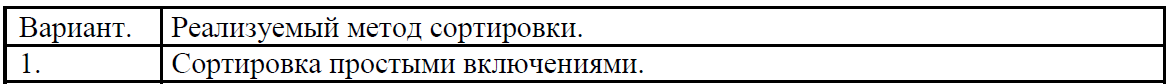
**Скріншот програми:**



В цьому завданні я розбив задачу сортування на меньші підзадачі та викликаючи функцію shellround для значень проміжків виведених сенджвіком я відсортував вхідний массив

**Завдання 3**

Написать программу сортировки [6] списка в соответствии с вариантом в таблице 3.



**Код програми:**

(defun ins(A B)

(cond

((null B) (list A))

((> A (car B)) (cons (car B) (ins A (cdr B))))

(T (cons A (ins (car B) (cdr B))))

)

)

(defun checksort(A)

(cond

((null (cdr A)) T)

((> (car A) (cadr A)) nil)

(T (checksort (cdr A)))

)

)

(defun T3(A)

(cond

((checksort A) A)

(T (T3 (ins (car A) (cdr A))))

)

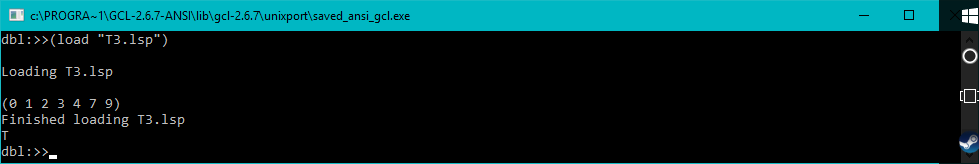
)

(print

(T3 '(1 9 2 7 4 3 0))

)

**Скріншот програми:**



В цьому завданні я використав сортування простим включенням щоб відсортувати масив. Для його реалізації я написав функції що робить один прохід сортування включення та перевіряє масив на відсортованість. Таким чином поки масив не відсортований будуть повторюватись проходи сортування включенням

**Завдання 4**

Написать программу объединения двух отсортированных списков в один. При этом порядок сортировки в списке-результате должен сохраняться.

**Код програми:**

(defun mrg (A B)

(cond

((null A) B)

((null B) A)

((< (car A) (car B)) (cons (car A) (mrg (cdr A) B)))

(T (cons (car B) (mrg A (cdr B))))

)

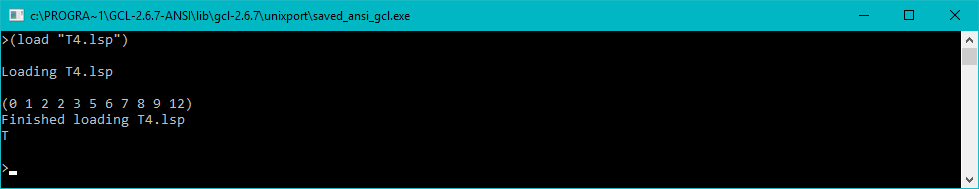
)

(print

(mrg '(0 2 3 5 9 12) '(1 2 6 7 8))

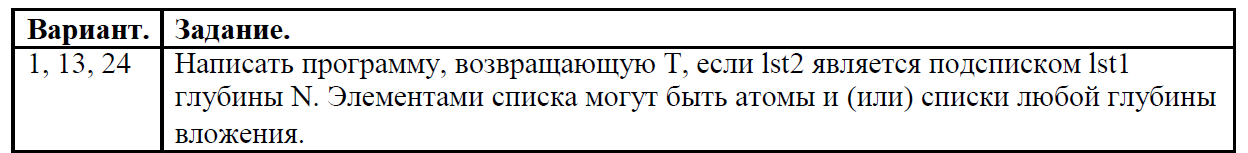
)

**Скріншот програми:**



В цьому завданні я написав функцію що перевіряє вхідні списки на порожність та у разі якщо жоден з них не порожній вибирає найменший елемент з двох та продовжує роботу з залишившимись списками

**Завдання 5**

Написать программу в соответствии с заданием из Таблицы 4. 

**Код програми:**

(defun exs (A B)

(cond

((null A) nil)

((not (atom (car A))) (exs (cdr A) B))

((= (car A) B) T)

(T (exs (cdr A) B))

)

)

(defun incl (A B)

(cond

((null B) T)

(T (and (exs A (car B)) (incl A (cdr B))))

)

)

(defun f (A B N)

(cond

((= N 0) (incl A B))

(T

(cond

((null A) nil)

((atom (car A)) (f (cdr A) B N))

(T (or (f (cdr A) B N) (f (car A) B (- N 1))))

)

)

)

)

(print

(f '(0 2 (3 (9 6) 4)(0) ((3))) '(9 6) 2)

)

(print

(f '(0 2 (3 (9 6) 4)(0) ((3))) '(3) 2)

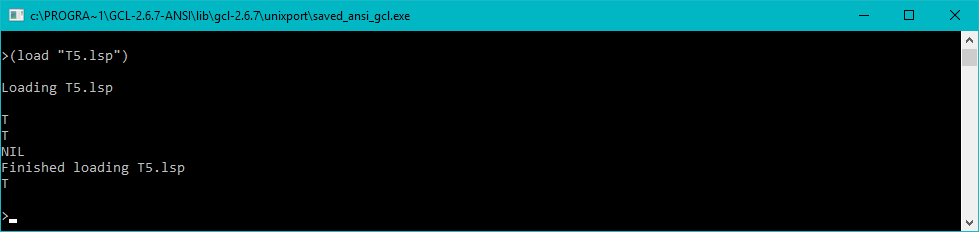
)

(print

(f '(0 2 (3 (9 6) 4)(0) ((3))) '(0) 2)

)

**Скріншот програми:**



В цьому завданні я написав функцію що спочатку рекурсивно спускається до необхідного рівня а потім перевіряє чи є данний список підсписком поточного

**Висновок:**

В цій лабораторній роботі я опанував базовий механізм виклику та опису рекурсивних функцій та рекурсивної обробки списків. Також я навчився писати рекурсивні реалізації алгоритмів сортувань на функціональній мові.