ЗАВДАННЯ
до лабораторних робіт з дисципліни “Програмування інтелектуальних інформаційних систем”.

Лабораторна 1. Мова штучного інтелекту

1. Опишіть, що відбувається при обчисленні таких висловлювань

(a) `(+ (- 5 1)  (+3 7))`

(b) `(list 1 (+  2 3))`

(c) `(if (listp 1) (+ 1 2) (+3 4))`

(d) `(list (and  (listp 3) t) (+ 1 2))`

2. Складіть за допомогою `cons` три різних вирази, що створюють `(a b c)`.

3. За допомогою `car` і `cdr` визначте функцію, яка повертає четвертий елемент списку.

4. Визначте функцію, приймаючу два аргументи і яка повертає найбільший з них.

5. Що роблять такі функції?

(a) `(defun enigma (x) (and (not (null x)) (or (null (car x)) (enigma (cdr x)))))`

(b) `(defun mystery (x y) (if (null y) nil (if (eql (car y) x) 0 (let ((z (mystery x (cdr y)))) (and z (+ z 1))))))`

6. Що може стояти на місці `x` в таких висловлюваннях?

```
(a)> (car (x (cdr '(a (b c) d))))
B
(b)> (x 13 (/ 1 0))
13
(c)> (x # 'list 1 nil)
(1)
```

7. Визначте функцію, яка перевіряє, чи є списком хоча б один елемент списку. Користуйтеся тільки тими операторами, які були згадані в третій лекції.

8. Запропонуйте ітеративне і рекурсивне визначення функції, яка:

(a) друкує кількість точок, що дорівнює заданому позитивному цілому числу;

(b) повертає кількість символів a в заданому списку.

9. Ваш товариш намагається написати функцію, яка підсумовує всі значення елементів списку, крім `nil`. Він написав дві версії такої функції, але жодна з них не працює. Поясніть, що не так в кожній з них, і запропонуйте коректну версію:

(a)
```
(defun summit (lst)
(remove nil lst)
(apply #’+ lst))
```

(b)
```
(defun summit (lst)
(let ((x (car lst)))
(if (null x)
(summit (cdr lst))
(+ x (summit (cdr lst))))))
```

10. Уявіть наступні списки у вигляді комірок:

(a) `(A b (c d))`

(b) `(A (b (c (d))))`

(c) `(((A b) c) d)`

(d) `(A (b. C). D)`

11. Напишіть свій варіант функції `union`, який зберігає порядок слідування елементів згідно з вихідними списками:

```
> (New-union '(a b c)' (b a d))
(A B C D)
```

12. Напишіть функцію, визначальну кількість повторень (з точки зору **eql**) кожного елемента в заданому списку і сортують їх по спадаючій народження:

```
> (occurrences '(a b a d a c d c a))
((A. 4) (C. 2) (D. 2) (B. 1))
```

13. Чому **(member '(a)' ((a) (b)))** повертає **nil?**

Лабораторна №2. Програмування складних структур на Lisp

1. Функція **pos +** приймає список і повертає новий, кожен елемент якого збільшений в порівнянні з вихідним на його положення в списку:

```
> (pos+ '(7 5 1 4))
(7 6 3 7)
```

Визначте цю функцію за допомогою: (a) рекурсії, (b) ітерації і (c) **mapcar**.

2. Після довгих років роздумів державна комісія прийняла постанову, згідно з яким **cdr** вказує на перший елемент списку, а **car** - на його залишок. Визначте наступні функції, що задовольняють цій постанові:

(a) **cons**
(b) **list1**
(c) **length** (для списків)
(d) **member** (для списків, без ключових параметрів)

3. Змініть програму таким чином, щоб вона створювала меншу кількість комірок. (Примітка: використовуйте точкові пари.) Нижче показаний подібний алгоритм для списків. Функція **compress** приймає список з атомів і повертає його стисле уявлення:

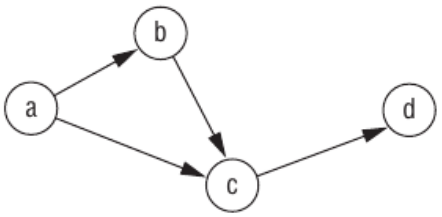
```
> (compress '(1 1 1 0 1 0 0 0 0 1))
((3 1) 0 1 (4 0) 1)
```

```
(defun compress (x)
     (if (consp x)
     (compr (car x) 1 (cdr x))
     x))
(defun compr (elt n lst)
     (if (null lst)
     (list (n-elts elt n))
     (let ((next (car lst)))
     (if (eql next elt)
     (compr elt (+ n 1) (cdr lst))
     (cons (n-elts elt n)
     (compr next 1 (cdr lst)))))))
(defun n-elts (elt n)
     (if (> n 1)
     (list n elt)
     elt))
```

4. Визначте функцію, що друкує заданий список в точкової нотації:

```
> (showdots '(a b c))
(A . (B . (C . NIL)))
NIL
```

5.Напішіте програму, яка шукає найбільш довгий шлях в мережі, що не містить повторень . Мережа може містити цикли.



6. Визначте функцію, повертаючу квадратний масив (масив з размерностями **(n n)**) на 90 градусів за годинниковою стрілкою:

```
> (quarter-turn #2A((a b) (c d)))
#2A((C A) (D B))
```

Вам буде потрібно **array-dimensions**:

```
(array-dimension array i)
```

Повертає **i**-ю розмірність масиву. Індексування починається з нуля.

7.Разберітесь з описом **reduce.**

```
(reduce function proseq
 &key key from-end start end initial-value)
```

Поведінка **reduce** описується в наступній таблиці, де **f** - функція, а елементами **proseq** є **a, b** і **c**:

from-end	*initial-value*	**эквивалент**
ложь	нет	$(f\ (f\ a\ b)\ c)$
ложь	да	$(f\ (f\ (f\ initial\text{-}value\ a)\ b)\ c)$
истина	нет	$(f\ a\ (f\ b\ c))$
истина	да	$(f\ a\ (f\ b\ (f\ c\ initial\text{-}value)))$

Якщо **proseq** складається з одного елемента і значення **initial-value** не надано, повертається сам елемент. Якщо **proseq** порожня і надано **initial-value**, воно повертається, але якщо **initial-value** не надано, функція викликається без аргументів. Якщо надані **key** і **initial-value**, **key** не застосовується до **initial-value**.

Потім з її допомогою визначьте:

(a) **copy-list**

(b) **reverse** (для списків)

8. Створіть структуру для дерева, кожен вузол якого крім деяких даних має трьох нащадків. Визначте:

(a) функцію, що копіює таке дерево (кожен вузол скопіюйте
ванного дерева не повинен бути еквівалентний вихідному з точки зору **eql**);

(b) функцію, приймаючу об'єкт і таке дерево і повертаючу істину, якщо цей об'єкт зустрічається (з точки зору **eql**) в поле даних хоча б одного вузла дерева.

9. Визначте функцію, яка будує з **BST**-дерева список його об'єктів, відсортоване від більшого до меншого.

10. Визначте **bst-adjoin1**. Функція працює аналогічно **bst-insert**, однак додає новий об'єкт лише в тому випадку, якщо він відсутній в наявному дереві.

11. Вміст будь-хеш-таблиці може бути представлено у вигляді асоціативного списку з елементами **(k. V)** для кожної пари ключ-значення. Визначте функцію, яка будує:

(a) хеш-таблицю по асоціативному списку;

(b) асоціативний список по хеш-таблиці.