

Стандартные функции обработки списков системы CommonLisp

Определение списка как структуры данных

В языках, предназначенных для программирования задач искусственного интеллекта, важную роль играют динамические структуры данных, называемые списками.

Главное достоинство списков состоит в том, что при их обработке операции вставки, замены и удаления элементов выполняются весьма просто.

Определение списка как структуры данных

Список - это упорядоченная последовательность, элементами которой являются атомы или списки (подсписки). Списки заключаются в круглые скобки, элементы списка разделяются пробелами. Открывающие и закрывающие скобки находятся в строгом соответствии.

Определение списка как структуры данных

Список всегда начинается с открывающей и заканчивается закрывающей скобкой. Список, который состоит из 0 элементов называют пустым и обозначают () или NIL. Пустой список — это атом.

Обработка списков

Списки — структуры данных с последовательным доступом, и поэтому основным средством их обработки является рекурсия.

Стандартные функции обработки списков

Стандартные функции обработки списков. Функции выборки.

N	Функция	Значение функции, примеры
1	(CAR S)	<p>1. S-список. CAR выдает первый элемент списка (CAR '(A B C D)) A</p> <p>2.S-точечная пара. CAR выдает левую часть точечной пары (CAR '((A . B) . C) (A . B)</p> <p>3. S-атом. CAR выдает значение атома, присвоенное ему специальными функциями присваивания (SETQ X 7) 7 (CAR 'X) 7</p> <p>В случае если у атома значение отсутствует, выдается имя атом</p>

Стандартные функции обработки списков. Функции выборки.

N	Функция	Значение функции, примеры
2	(CDR S)	<p>1. S-список. CDR возвращает список без первого элемента (CDR '(A B C D)) (B C D)</p> <p>2. S-точечная-пара. CDR возвращает правую часть точечной пары (на верхнем уровне) (CDR '((A . B) . C)) C</p> <p>3. S-атом. CDR возвращает список свойств атома (PUT 'JOE 'SEX 'MALE) MALE (CDR 'JOE) ((SEX . MALE))</p>

Стандартные функции обработки списков. Функции выборки.

N	Функция	Значение функции, примеры
3	(LAST L)	LAST возвращает список, в котором убраны все элементы, кроме последнего. (LAST '(A B C D)) (D) (LAST '(A B C . D)) (C . D) (LAST 'A) NIL

Стандартные функции обработки списков. Функции выборки.

N	Функция	Значение функции, примеры
4	(NTHCDR N L) Элементы списка нумеруются с нуля.	NTHCDR возвращает N-й CDR от списка L. (NTHCDR 0 '(A B C D)) (A B C D) (NTHCDR 1 '(A B C D)) (B C D) (NTHCDR 2 '(A B C D)) (C D) (NTHCDR 5 '(A B C D)) NIL (NTHCDR 2 '(A B . C)) C

Стандартные функции обработки списков. Функции выборки.

N	Функция	Значение функции, примеры
5	(NTH N L)	NTH возвращает N-й элемент списка (отсчет ведется от нуля). (NTH 0 '(A B C D)) A (NTH 3 '(A B C D)) D (NTH 4 '(A B C D)) NIL

Стандартные функции обработки списков. Функции конструирования.

N	Функция	Значение функции, примеры
1	(CONS S1 S2)	<p>1. S2-список. CONS возвращает список S2, перед первым элементом которого добавлен аргумент S1.</p> <pre>\$ (CONS 'A '(B C D)) (A B C D)</pre> <p>2. S2-атом. CONS возвращает точечную пару, левая часть которой S1, а правая - S2.</p> <pre>\$ (CONS 'A 'B) (A . B)</pre>
2	(LIST S1 S2 ... Sn)	<p>LIST конструирует и возвращает список, из элементов S1, S2,...Sn. При отсутствии аргументов возвращает NIL</p> <pre>\$ (LIST 'A 'B 'C 'D) (A B C D) \$ (LIST 'A '(B C) 'D) (A (B C) D) \$ (LIST) NIL</pre>

Стандартные функции обработки списков. Функции конструирования.

N	Функция	Значение функции, примеры
3	(APPEND L1 L2 ...LN)	APPEND конструирует и возвращает список из элементов списков L1, L2 ... Ln \$ (APPEND '(A B C) '(D E F)) (A B C D E F) \$ (SETQ FOO '(1 2 3)) (1 2 3) \$ (APPEND '(A B C) FOO '(K L M)) (A B C 1 2 3 K L M)
4	(REVERSE L)	REVERSE возвращает реверсированный список L (элементы списка L в обратном порядке) \$ (REVERSE '(A B C D E)) (E D C B A) \$ (REVERSE '(A B C) '(1 2 3)) (C B A 1 2 3) \$ (REVERSE '(A B C) 'D) (A B C . D)

Стандартные функции обработки списков. Функции конструирования.

N	Функция	Значение функции, примеры
5	(LENGTH S)	<p>1.S-список. LENGTH возвращает число элементов в списке L \$ (LENGTH '(A B C D E)) 5 </p> <p>2.S-атом. LENGTH возвращает число символов в атоме \$ (LENGTH 'TIME) 4</p> <p>3.S-число. LENGTH возвращает количество байтов занимаемых числом в памяти \$ (LENGTH -13) 1</p>

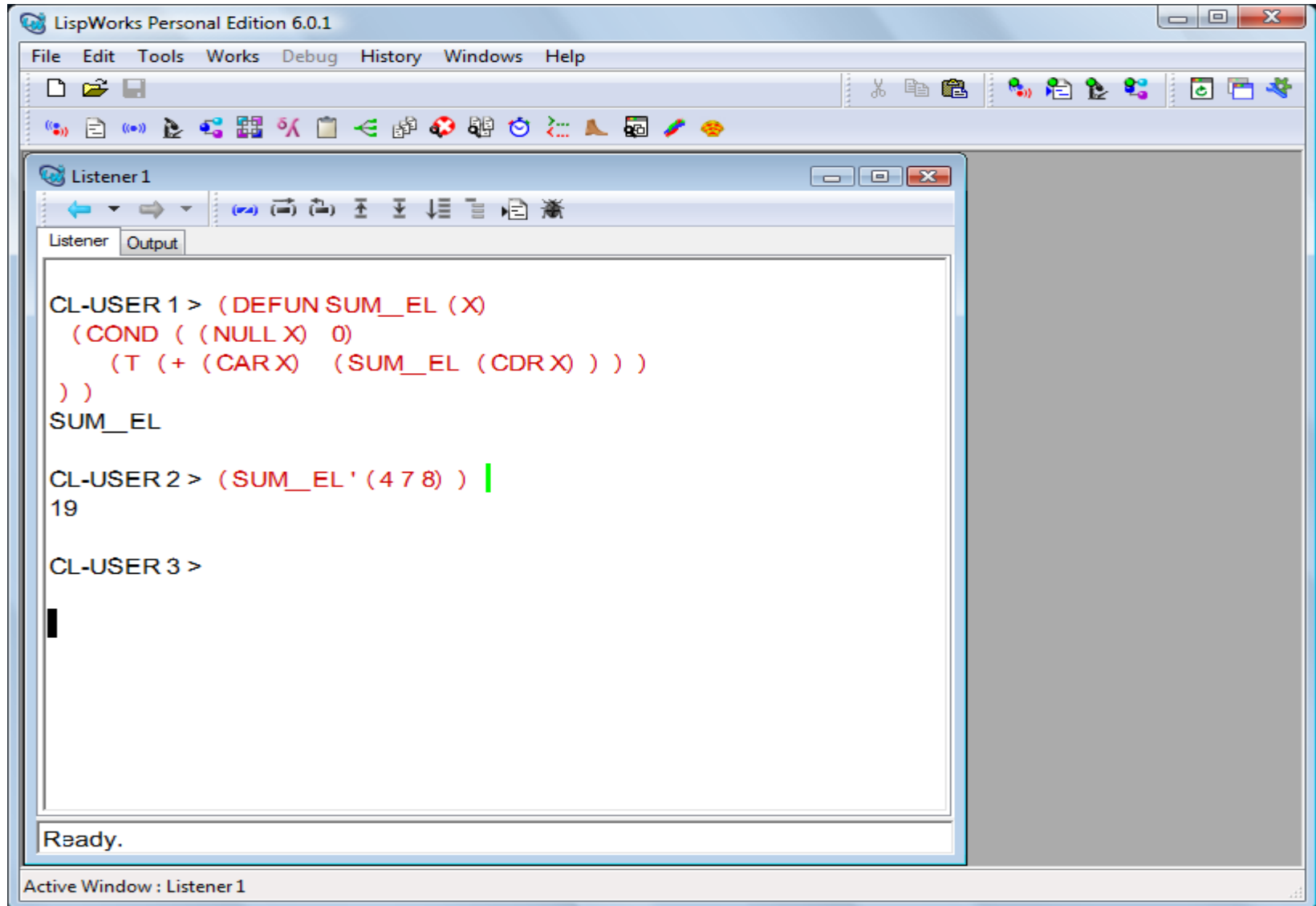
Построение новых функций в среде Common Lisp

Примеры функций обработки
СПИСКОВ

Пример1. Функция, определяющая сумму элементов
списка

```
(DEFUN SUM_EL(X)  
  (COND ((NULL X) 0)  
        (T (+ (CAR X) (SUM_EL (CDR X)))))  
))
```

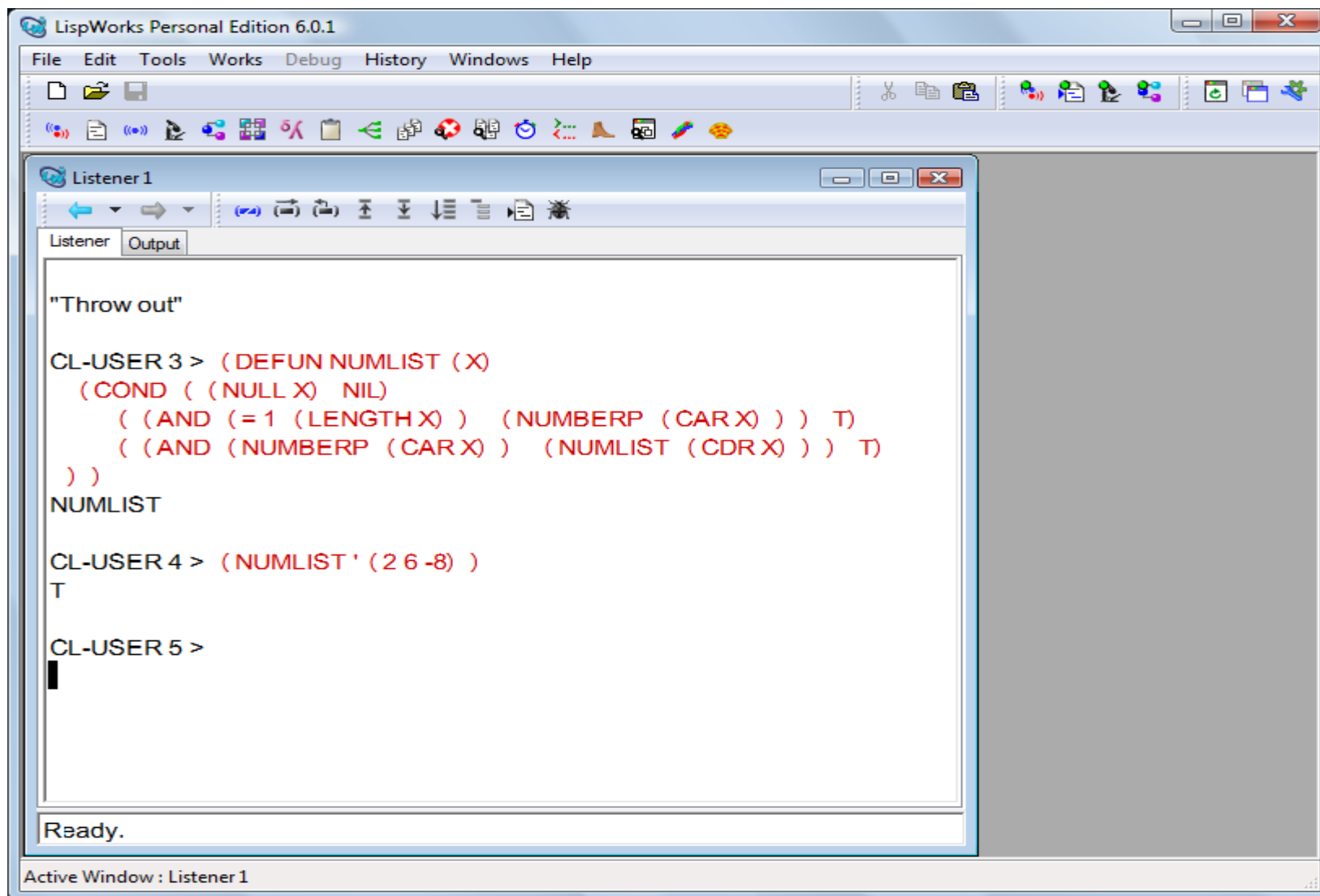

Пример1. Экран Common LISP с примером вызова функции, определяющей сумму элементов списка



Пример2. Функция, определяющая является ли
заданный список списком целых чисел

```
(DEFUN NUMLIST(X)
  (COND ((NULL X) NIL)
        ((AND (= 1 (LENGTH X)) (NUMBERP (CAR X)))
         T)
        ((AND (NUMBERP (CAR X)) (NUMLIST (CDR
X)))) T)
))
```

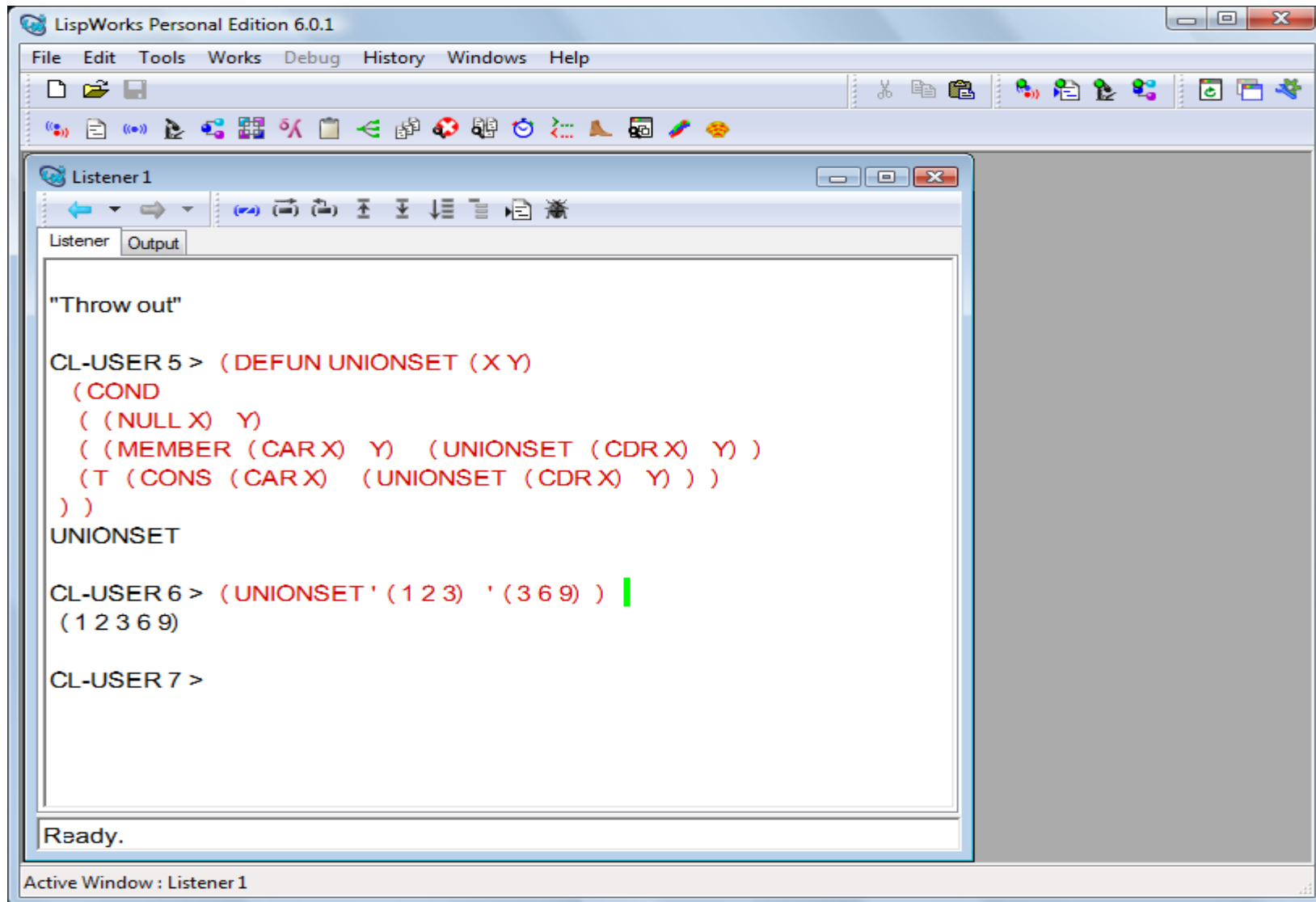
Экран Common LISP с примером вызова функции, определяющей является ли заданный список списком целых чисел



Пример3. Функция, определяющая объединение списков как множеств

```
(DEFUN UNIONSET(X Y)
  (COND
    ((NULL X) Y)
    ((MEMBER (CAR X) Y) (UNIONSET (CDR X) Y))
    (T (CONS (CAR X) (UNIONSET (CDR X) Y))))
  ))
```

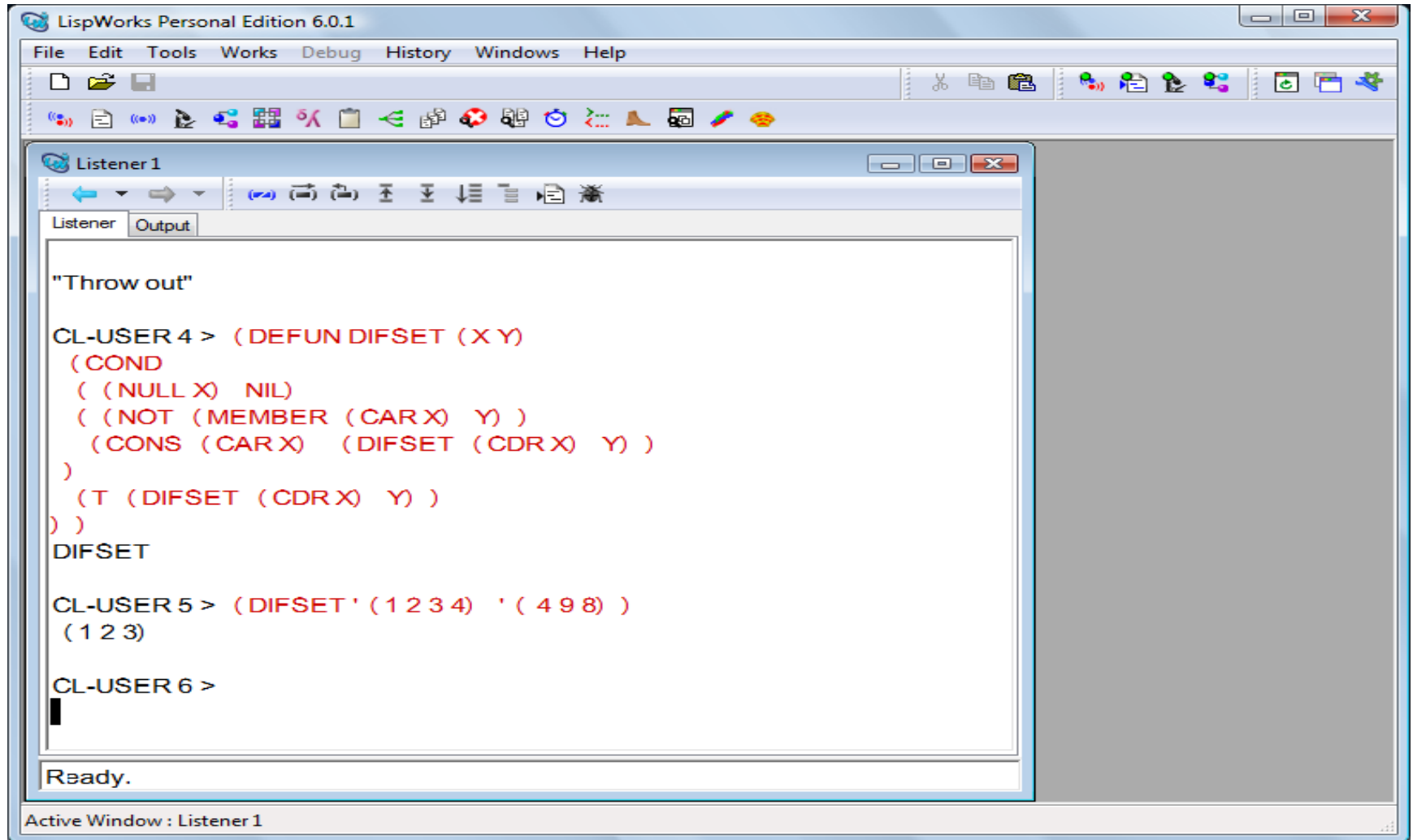
Экран Common LISP с примером вызова функции, определяющей объединение множеств



Функция, определяющая разность двух множеств

```
(DEFUN DIFSET(X Y)
  (COND
    ((NULL X) NIL)
    ((NOT (MEMBER (CAR X) Y))
     (CONS (CAR X) (DIFSET (CDR X) Y)))
    )
  (T (DIFSET (CDR X) Y)))
))
```

Выполнение функции, определяющей разность двух множеств



The screenshot shows the LispWorks Personal Edition 6.0.1 window. The 'Listener 1' pane is active, displaying the following text:

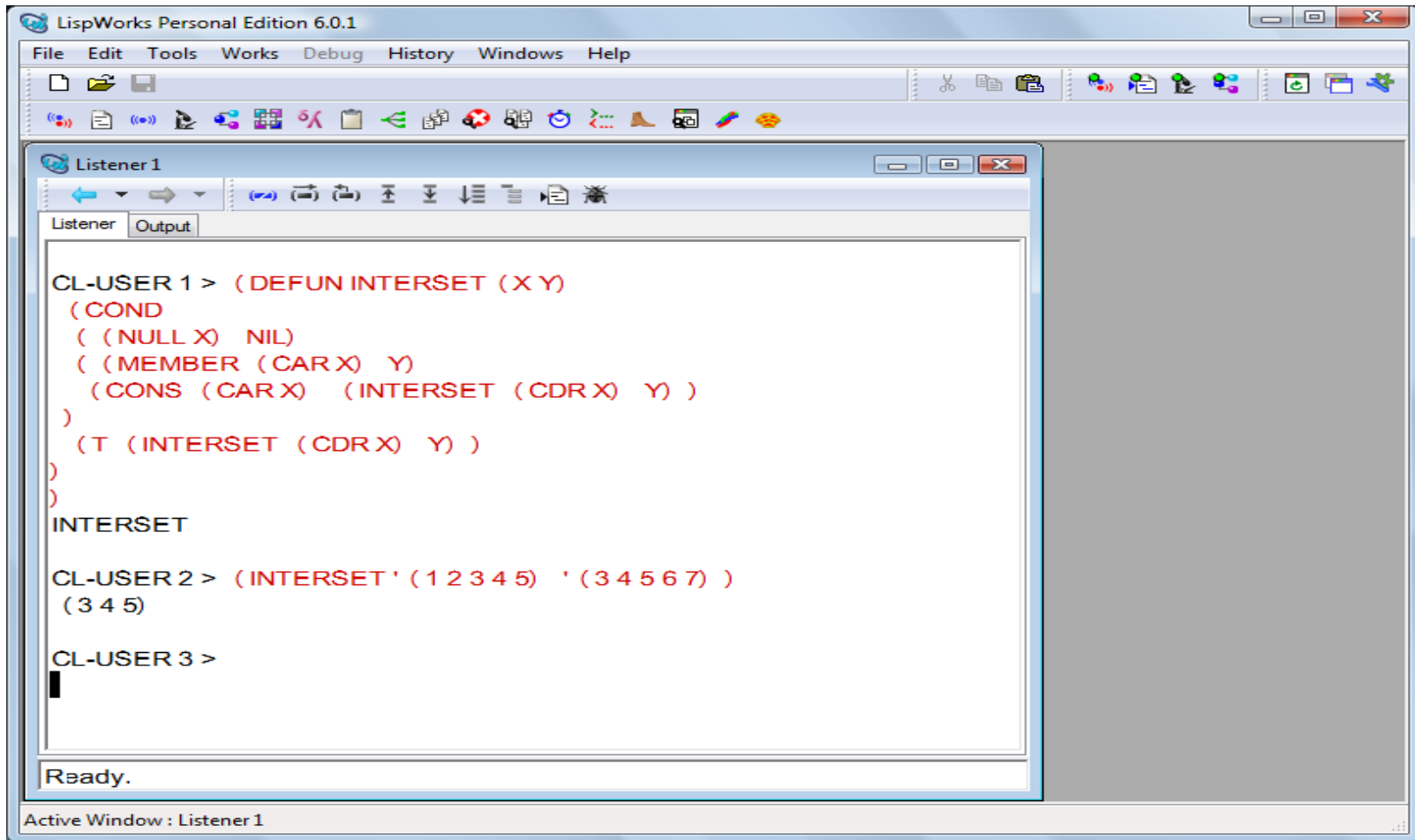
```
"Throw out"  
  
CL-USER 4 > (DEFUN DIFSET (X Y)  
  (COND  
    ((NULL X) NIL)  
    ((NOT (MEMBER (CAR X) Y))  
     (CONS (CAR X) (DIFSET (CDR X) Y))  
    )  
    (T (DIFSET (CDR X) Y))  
  ) )  
DIFSET  
  
CL-USER 5 > (DIFSET '(1 2 3 4) '(4 9 8))  
(1 2 3)  
  
CL-USER 6 >  
Ready.
```

The status bar at the bottom indicates 'Active Window : Listener 1'.

Функция, определяющая пересечение двух множеств

```
(DEFUN INTERSET(X Y)
  (COND
    ((NULL X) NIL)
    ((MEMBER (CAR X) Y)
     (CONS (CAR X) (INTERSET (CDR X) Y)))
    )
  (T (INTERSET (CDR X) Y)))
)
```


Выполнение функции, определяющей пересечение двух множеств



The screenshot shows the LispWorks Personal Edition 6.0.1 application window. The main window has a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help) and a toolbar. A 'Listener 1' window is open, displaying the following Lisp code and output:

```
CL-USER 1 > (DEFUN INTERSET (X Y)
  (COND
    ( (NULL X) NIL)
    ( (MEMBER (CAR X) Y)
      (CONS (CAR X) (INTERSET (CDR X) Y) )
    )
    (T (INTERSET (CDR X) Y) )
  )
)
INTERSET

CL-USER 2 > (INTERSET '(1 2 3 4 5) '(3 4 5 6 7) )
(3 4 5)

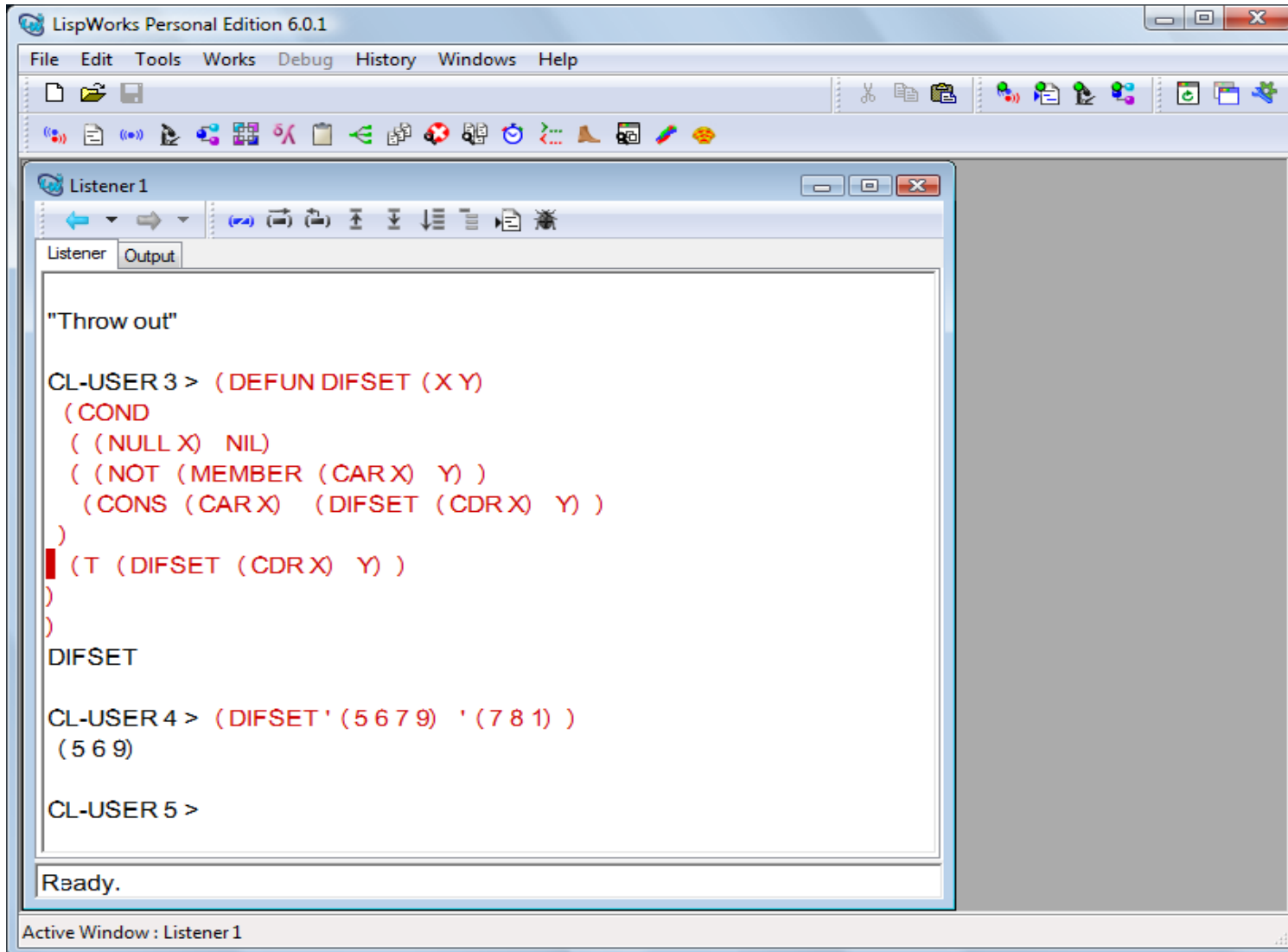
CL-USER 3 >
Ready.
```

The status bar at the bottom indicates 'Active Window : Listener 1'.

Функция, определяющая разность двух множеств

```
(DEFUN DIFSET(X Y)
  (COND
    ((NULL X) NIL)
    ((NOT (MEMBER (CAR X) Y))
     (CONS (CAR X) (DIFSET (CDR X) Y)))
    )
  (T (DIFSET (CDR X) Y))
)
```

Выполнение функции, определяющей разность двух множеств



The screenshot shows the LispWorks Personal Edition 6.0.1 application window. The main window has a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help) and a toolbar. A 'Listener 1' window is open, displaying the following text:

```
"Throw out"  
  
CL-USER 3 > (DEFUN DIFSET (X Y)  
  (COND  
    ((NULL X) NIL)  
    ((NOT (MEMBER (CAR X) Y))  
     (CONS (CAR X) (DIFSET (CDR X) Y))  
    )  
  )  
  (T (DIFSET (CDR X) Y))  
)  
DIFSET  
  
CL-USER 4 > (DIFSET ' (5 6 7 9) ' (7 8 1) )  
(5 6 9)  
  
CL-USER 5 >  
  
Ready.
```

The status bar at the bottom indicates 'Active Window : Listener 1'.

Функция, определяющая декартово произведение двух множеств. Функция PRO

```
(DEFUN PRO(X Y)
```

```
(COND
```

```
((NULL Y) NIL)
```

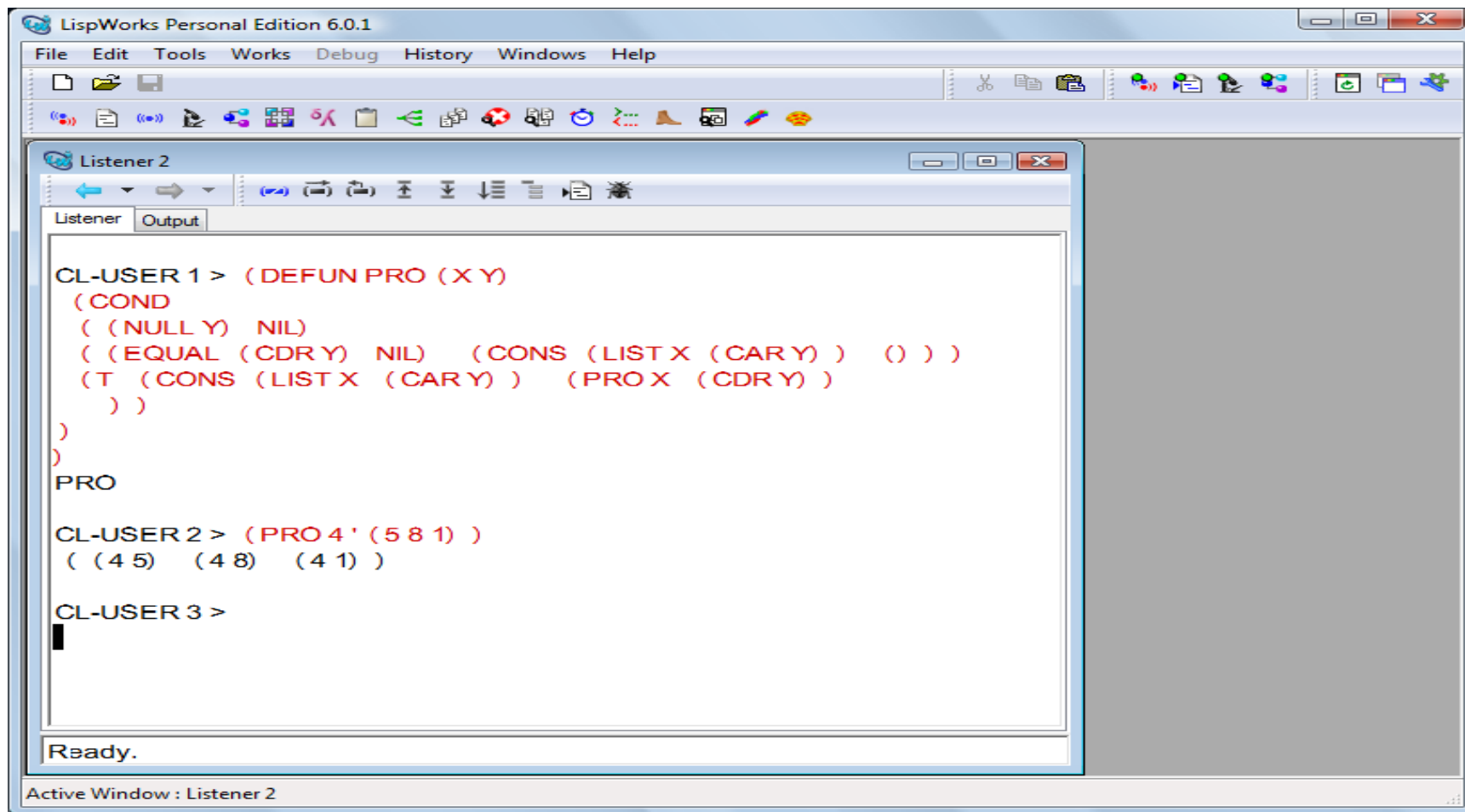
```
((EQUAL (CDR Y) NIL) (CONS (LIST X (CAR Y)) ()))
```

```
(T (CONS (LIST X (CAR Y)) (PRO X (CDR Y))  
      ))
```

```
)
```

```
)
```

Выполнение функции PRO



The screenshot shows the LispWorks Personal Edition 6.0.1 window. The 'Listener 2' pane is active, displaying the following code and output:

```
CL-USER 1 > (DEFUN PRO (X Y)
  (COND
    ((NULL Y) NIL)
    ((EQUAL (CDR Y) NIL) (CONS (LIST X (CAR Y)) ()))
    (T (CONS (LIST X (CAR Y)) (PRO X (CDR Y))
      ) )
  )
)
PRO

CL-USER 2 > (PRO 4 '(5 8 1) )
( (4 5) (4 8) (4 1) )

CL-USER 3 >
█

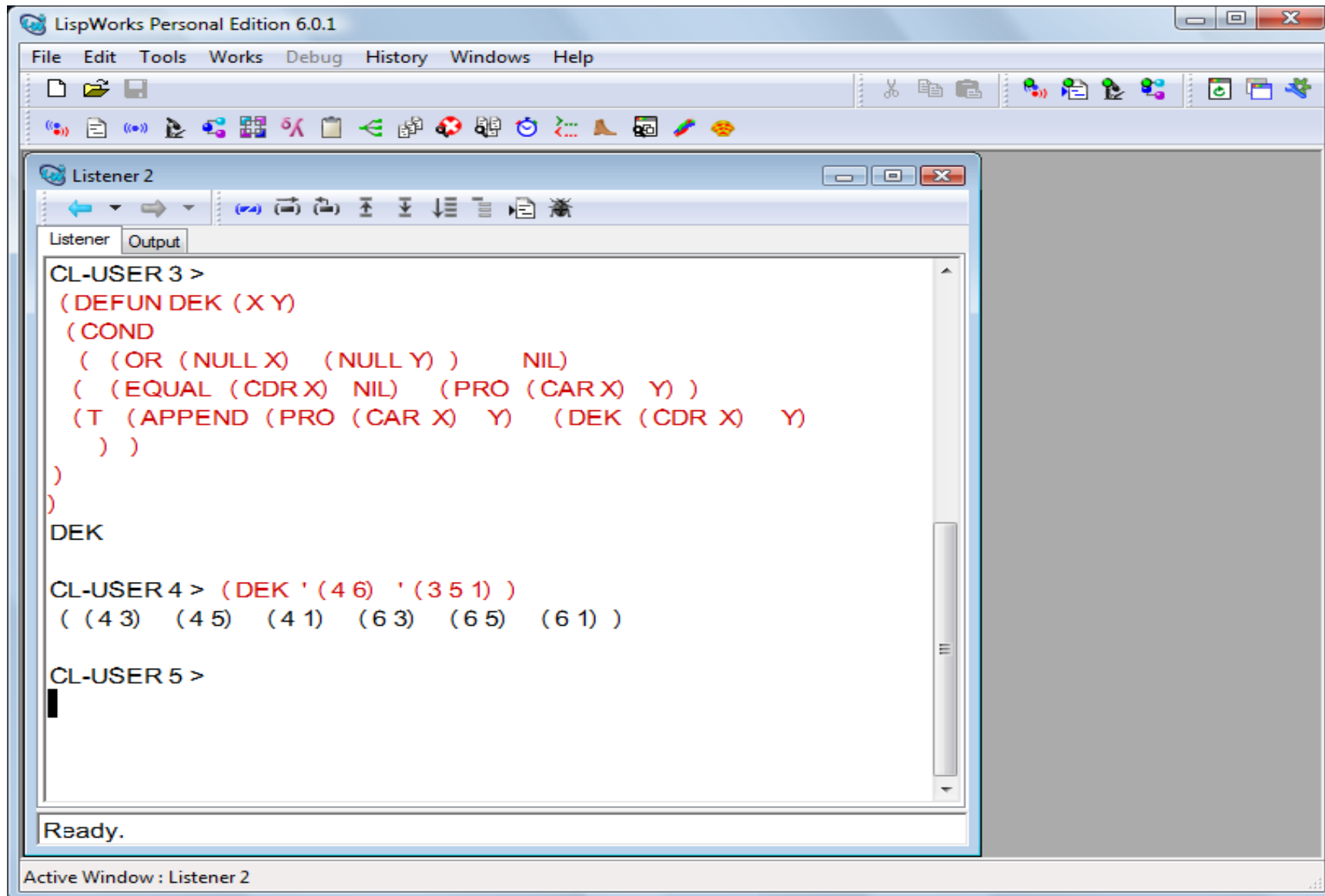
Ready.
```

The status bar at the bottom indicates 'Active Window : Listener 2'.

Функция, определяющая декартово произведение двух множеств. Функция DEK

```
(DEFUN DEK(X Y)
  (COND
    ( (OR (NULL X) (NULL Y))      NIL)
    ( (EQUAL (CDR X) NIL) (PRO (CAR X) Y))
    (T (APPEND (PRO (CAR X) Y) (DEK (CDR X)
      Y)
      ) )
  )
)
```

Выполнение функции DEK



The screenshot shows the LispWorks Personal Edition 6.0.1 application window. The main window is titled "Listener 2" and contains a text area with the following Lisp code and its output:

```
CL-USER 3 >  
(DEFUN DEK (X Y)  
  (COND  
    ( (OR (NULL X) (NULL Y))      NIL)  
    ( (EQUAL (CDR X) NIL) (PRO (CAR X) Y) )  
    (T (APPEND (PRO (CAR X) Y) (DEK (CDR X) Y) )  
      ) )  
  )  
)  
DEK  
  
CL-USER 4 > (DEK '(4 6) '(3 5 1) )  
( (4 3) (4 5) (4 1) (6 3) (6 5) (6 1) )  
  
CL-USER 5 >  
█
```

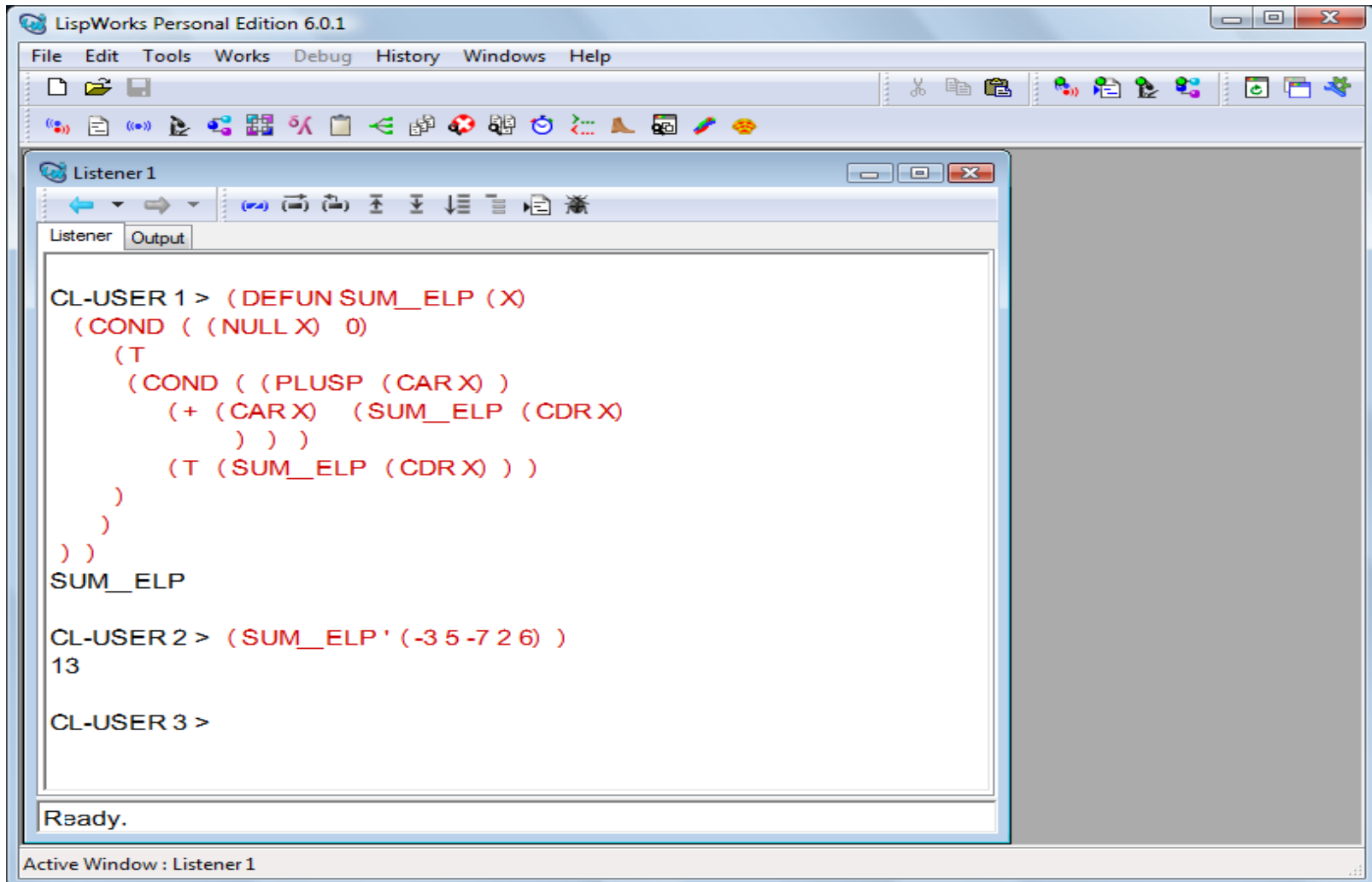
At the bottom of the window, the status bar indicates "Ready." and "Active Window : Listener 2".

Функция, определяющая сумму положительных элементов списка

```
(DEFUN SUM_ELP(X)
  (COND ((NULL X) 0)
        (T
         (COND ((PLUSP (CAR X))
                  (+ (CAR X) (SUM_ELP (CDR X))
                     ) ) )
         (T (SUM_ELP (CDR X)))
        )
  )
)
```

-->SUM_ELP

Выполнение функции, определяющей сумму положительных элементов списка



The screenshot shows the LispWorks Personal Edition 6.0.1 application window. The main window has a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help) and a toolbar. A 'Listener 1' window is open, displaying the following Lisp code and output:

```
CL-USER 1 > (DEFUN SUM_ELP (X)
  (COND ( (NULL X) 0)
    (T
      (COND ( (PLUSP (CAR X) )
        (+ (CAR X) (SUM_ELP (CDR X) )
          ) ) )
      (T (SUM_ELP (CDR X) ) ) )
  )
)
SUM_ELP

CL-USER 2 > (SUM_ELP ' (-3 5 -7 2 6) )
13

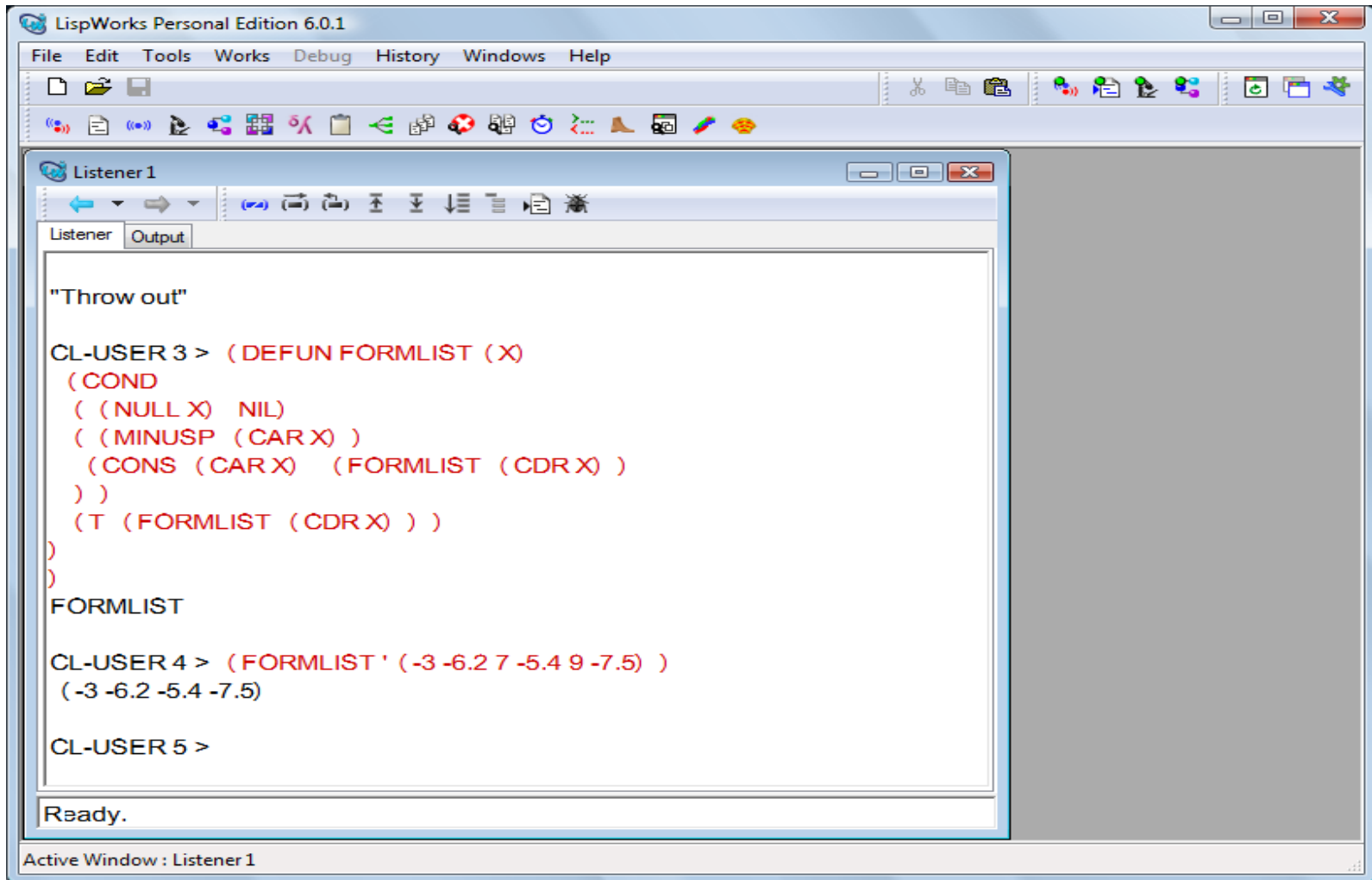
CL-USER 3 >
```

The status bar at the bottom indicates 'Active Window : Listener 1'.

**Функция, формирующая из исходного
списка список отрицательных элементов**

```
(DEFUN FORMLIST(X)
  (COND
    ((NULL X) NIL)
    ((MINUSP (CAR X))
      (CONS (CAR X) (FORMLIST (CDR X)))
    ))
    (T (FORMLIST (CDR X)))
  )
)
```

Выполнение функции, формирующей из исходного списка список отрицательных элементов



The screenshot shows the LispWorks Personal Edition 6.0.1 application window. The main window has a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help) and a toolbar. A 'Listener 1' window is open, displaying the following text:

```
"Throw out"  
  
CL-USER 3 > (DEFUN FORMLIST (X)  
  (COND  
    ( (NULL X) NIL)  
    ( (MINUSP (CAR X))  
      (CONS (CAR X) (FORMLIST (CDR X)) )  
    ) )  
  (T (FORMLIST (CDR X)) ) )  
)  
  
FORMLIST  
  
CL-USER 4 > (FORMLIST ' (-3 -6.2 7 -5.4 9 -7.5) )  
(-3 -6.2 -5.4 -7.5)  
  
CL-USER 5 >  
  
Ready.
```

The status bar at the bottom indicates 'Active Window : Listener 1'.