Основы использования и работы в прикладной компьютерной программе (системе компьютерной алгебры) Maxima.

Задания вариативной самостоятельной работы.

Часть 2. Задание 3.5.

Справочник по формулам Maxima, используемых при работе со списками

Список в Maxima – это упорядоченная совокупность произвольных объектов. Чтобы задать список, достаточно записать его элементы через запятую и ограничить запись квадратными скобками

```
list1:[a,b,11];
                        [a,b,11]
```

Элементом списка может быть любой объект, в том числе и другой список

```
list2:[a,b,[c,d],e,f];
                    [a,b,[c,d],e,f]
```

Список может быть пустым

```
list3:[];
                             []
```

Или состоять из одного элемента

```
list4:[77];
                            [77]
```

Ссылка на элемент списка производится так:

```
list1[2];
                              b
list1[2]:c;
                              c
list1;
                        [a,c,11]
list2[3];
                          [c,d]
list2[3][2];
                              d
```

- **Функция** *length*

возвращает длину списка

```
length([a,b,[c,d],e,f]);
                              5
```

- **Функция** *part*

позволяет выделить тот или иной элемент, часть списка

```
part([a,b,c],2);
                          b
part([a,[b,c],d],2);
                        [b,c]
```

Если список вложенный, нужно писать так:

```
part([a,b,[c,d],e,f],3,2);
                          d
```

Следует подчеркнуть, что при присвоении списков

```
list1:[a,b,c]$
list2:list1;
                       [a,b,c]
```

отнюдь не создается копия списка «List1», просто переменная «List2» становится еще одним указателем на *тот же самый* список. Поэтому

```
list1[3]:d$
list2;
                       [a,b,d]
```

- **Функция** *copylist*

создает «настоящую» копию списка

```
list1:[a,b,c]$
list3:copylist(list1);
                       [a,b,c]
list1[3]:d$
list3;
                       [a,b,c]
```

- **Функция** *makelist*

позволяет создавать списки и допускает 2 варианта синтаксиса

```
makelist(i^3,i,1,3);
                      [1,8,27]
```

(здесь реализуется цикл по переменной «i» в пределах от 1 до 3), либо

```
makelist(x=i^2,i,[c,d,e]);
              [x=c^2, x=d^2, x=e^2]
```

(здесь переменная «i» пробегает все элементы заданного списка)

- **Функция** *append*

позволяет склеить два списка

```
append([a,b],[c,d]);
                    [a,b,c,d]
list1:[a]$ list2:[b,c]$
append(list1,list2);
                     [a,b,c]
```

- **Функция** *cons*

позволяет добавлять элемент в начало списка

```
cons(x,[a,b]);
                     [x,a,b]
```

- **Функция** *endcons*

позволяет добавлять элемент в конец списка

```
endcons(x,[a,b]);
                     [a,b,x]
```

- **Функция** *reverse*

меняет порядок элементов в списке на обратный

```
reverse(a,b,[c,d],e,f);
                  [f,e,[c,d],b,a]
```

- **Функция** *sort*

упорядочивает элементы списка

```
sort([3,a,-1,x,b,0]);
                [-1,0,3,a,b,x]
```

сначала идут числа (в порядке возрастания), затем индификаторы (в алфавитном порядке)

- **Функция** *sublist*

составляет список из тех элементов исходного списка, для которых заданная логическая функция возвращает значение «true»

```
sublist([3,a,-1,x,b,0],numberp);
                     [3,-1,0]
```

(функция «numberp» выдает «true» для чисел и «false» во всех остальных случаях). При этом может использоваться логическая функция, определенная пользователем.

```
f(x):=is(x>5)$
sublist([7,3,6,4,5],f);
                      [7,6]
```

- **Функция** *member*

возвращает «true», если ее первый аргумент является элементом заданного списка, и «false» в противном случае

```
list1:[a,b,[c,d],e,f]$
member(b,list1);
                        true
member(c,list1);
                       false
member([c,d],list1);
                        true
```

- **Функция** *first*

выделяет первый элемент списка

```
first([a,b,c]);
                          a
```

- **Функция** *rest*

выделяет остаток после удаления первого элемента списка

```
rest(a,b,c);
                       [b,c]
```

- **Функция** *last*

выделяет последний элемент списка

```
last([a,b,c]);
                          c
```

- **Функция** *map*

применяет заданную функцию к каждому элементу списка

```
map(sin,[-1,0,1]);
                [-sin(1),0,sin(1)]
```

при этом может использоваться как стандартная функция, так и функция, определенная пользователем

```
f(x):=2*x$ map(f,[1,2,3]);
                      [2,4,6]
```

- **Функция** *apply*

применяет заданную функцию ко всему списку (список становится списком аргументов функции). Например, если Вы соорудили список, состоящий из чисел:

```
list1:[33,-22.11]$
```

то, чтобы найти максимальное или минимальное число, надо вызывать функцию «max» или «min». Однако, обе функции в качестве аргумента

ожидают несколько чисел, а не список, составленный из чисел. применять подобные функции к спискам и позволяет функция «apply»:

```
apply(max,list1);
                          33
apply(min,list1);
                         -22
```