The Euclidean algorithm

(in various versions)

Reference: Ziegenbalg J.: Algorithmen von Hammurapi bis Gödel,
4. Auflage, Springer-Spektrum, Wiesbaden 2016, Section 3.2.2

1 The Euclidean algorithm by iterated subtraction

```
(%i1) Euclid_sub(a0, b0):=
        block([a : a0, b : b0],
          while not(a*b = 0) do
            /* while neither  a  nor  b  is zero */
           if a > b then a : a-b else b : b-a ,
          return(a+b) )       /* now, one of the summands is zero */  $ ;

(%i2) Euclid_sub(136,60);
(%o2) 4
```

Remark: Actually, using "return" in the above version is redundant. Just writing
a+b instead of return(a+b)
will do, because a+b is the last statement in Euclid and the result
of this call will be returned as the value of Euclid.
(The main function of the "return" command is to force the exit from the
processing of a function with return's parameter as the function value.)

Therefore, in the next examples using "return" as the last call will be avoided.

Also, in the next version the subtraction process is made visible by introduction
of the global variable "verbose". If the value of verbose is "true" then
intermediate values are printed, otherwise only the result is returned.

```
(%i3) verbose : true   /* global control variable */   $ ;

(%i4) Euclid_sub_verbose(a0, b0):=
        block([a : a0, b : b0],
          while not(a*b = 0) do
            (if a > b then a : a-b else b : b-a ,
             if verbose then print(a, "  ", b) ),
          a+b )  $ ;
```

```
(%i5) Euclid_sub_verbose(136, 60);
      76    60
      16    60
      16    44
      16    28
      16    12
      4     12
      4     8
      4     4
      4     0
(%o5) 4
```

2 The Euclidean algorithm by iterated (integer-) division

At first, some preparatory examples on integer division and equality in Maxima (quotient, mod, is, =)

```
(%i6) quotient(17, 5);
(%o6) 3

(%i7) mod(17, 5);
(%o7) 2

(%i8) 2+3=5;
(%o8) 5=5

(%i9) is(2+3=5);
(%o9) true

(%i10) quotient(17,5)*5+mod(17,5) = 17;
(%o10) 17=17

(%i11) is(quotient(17,5)*5+mod(17,5) = 17);
(%o11) true

(%i12) verbose : false ;
(verbose) false

(%i13) Euclid_div(a0, b0) :=
         block([a : a0, b : b0],
           while not(a*b = 0) do
             (if verbose then print(a, "  ", b),
              if a >= b then a : mod(a, b) else b : mod(b, a) ),
           return(a+b) ) $ ;

(%i14) Euclid_div(136,60);
(%o14) 4
```

```
(%i15) verbose : true;
(verbose) true

(%i16) Euclid_div(8765432, 2345678);
       8765432    2345678
       1728398    2345678
       1728398    617280
       493838     617280
       493838     123442
       70     123442
       70     32
       6     32
       6     2
(%o16) 2
```

3 The Euclidean algorithm by recursion

Two recursive versions are given:
first version: in the "subtraction" form
second version: in the "(integer) division" form

```
(%i17) Euclid_sub_rec(a, b) :=
        (if verbose then print(a, "  ", b),
         if a=0 then b
           else if b=0 then a
                 else if a>b then Euclid_sub_rec(a-b,b)
                       else Euclid_sub_rec(a, b-a) ) $ ;

(%i18) verbose : true;
(verbose) true

(%i19) Euclid_sub_rec(136,60);
       136    60
       76    60
       16    60
       16    44
       16    28
       16    12
       4    12
       4    8
       4    4
       4    0
(%o19) 4
```

```
(%i20) Euclid_div_rec(a, b) :=
         (if verbose then print(a, "  ", b),
          if a=0 then b
            else if b=0 then a
                 else if a>b then Euclid_div_rec(mod(a,b), b)
                      else Euclid_div_rec(a, mod(b,a)) ) $ ;

(%i21) verbose:true;
(verbose) true

(%i22) Euclid_div_rec(136,60);
       136    60
       16    60
       16    12
       4    12
       4    0
(%o22) 4
```