```
> # Contrôle du 13/11/2009
> #EXO 1 :
> somprem := proc(m,n)
  local S,i;
  S:=0;
  for i from m to n do
  if isprime(i)=true then
  S:=S+i;
  printf("i=%d\n",i);
  end if;
  end do;
  return S;
  end proc;
```

somprem := **proc**(*m*, *n*)
local *S*, *i*;
S := 0;
for *i* **from** *m* **to** *n* **do**
if isprime(*i*) = *true* **then**
S := *S* + *i*;
printf("i=%d
" , *i*)
end if
end do;
return *S*
end proc

```
> somprem(2,11);
i=2
i=3
i=5
i=7
i=11
```



```
> #EXO 2 :
> #1
> somdiv := proc(n)
  local S,i;
  S:=0;
  for i from 1 to n do
  if (irem(n,i)=0) then
  S:=S+i;
  end if;
  end do;
  return S;
  end proc;
```

somdiv := **proc**(*n*)

```
local S, i;
    S := 0; for i to n do if irem(n, i) = 0 then S := S + i end if end do; return S
end proc
> somdiv(3);
                                        4
> somdiv(5);
                                        6
> somdiv(4);
                                        7
> #2
> parfait := proc(n)
  if 2*n=somdiv(n) then return true;
  else return false;
  end if;
  end proc;
    parfait := proc(n) if 2*n = somdiv(n) then return true else return false end if end proc
> parfait(6);
                                      true
> parfait(5);
                                      false
> somdiv(5);
                                        6
> #3
> listeparfaits := proc(n)
  local L,i;
  L:=[];
  for i from 1 to n do
  if parfait(i)=true then
  L:=[op(L),i];
  end if;
  end do;
  return L;
  end proc;
listeparfaits := proc(n)
local L, i;
    L := [ ]; for i to n do if parfait(i) = true then L := [op(L), i] end if end do; return L
end proc
> listeparfaits(5);
                                      [ ]
> listeparfaits(6);
                                     [6]
> listeparfaits(12);
                                     [6]
> listeparfaits(100);
```

[6, 28]

```
> #EXO 3 :
> #1
> nbdiv:=proc(n)
  local S,i;
  S:=0;
  for i from 1 to n do
  if irem(n,i)=0 then S:=S+1;end if;
  end do;
  return S;
  end proc;
```

nbdiv := **proc**(*n*)

local *S*, *i*;

S := 0; **for** *i* **to** *n* **do if** irem(*n*, *i*) = 0 **then** *S* := *S* + 1 **end if end do**; **return** *S*

end proc

```
> nbdiv(5);
```

2

```
> nbdiv(6);
```

4

```
> #2
> plouton:=proc(N)
  local L,i,j,flag;
  L:=[];
  for i from 1 to N do
  flag:=true;
  j:=1;
  while (j<=i-1) do
  if nbdiv(j)>=nbdiv(i) then flag:=false; end if;
  j:=j+1;
  end do;
  if flag=true then L:=[op(L),i]; end if;
  end do;
  return L;
  end proc;
```

plouton := **proc**(*N*)

local *L*, *i*, *j*, *flag*;

L := [ ];

for *i* **to** *N* **do**

flag := *true*;

j := 1;

while *j* ≤ *i* − 1 **do if** nbdiv(*i*) ≤ nbdiv(*j*) **then** *flag* := *false* **end if**; *j* := *j* + 1 **end do**;

if *flag* = *true* **then** *L* := [op(*L*), *i*] **end if**

end do;

return *L*

```
end proc
> plouton(6);
```

[1, 2, 4, 6]

```
> plouton(12);
```

[1, 2, 4, 6, 12]

```
> #EXO 4 :
> somdivpropres:=proc(n)
  local S,i;
  S:=0;
  for i from 1 to n-1 do
  if irem(n,i)=0 then S:=S+i; end if;
  end do;
  return S;
  end proc;
```

somdivpropres := **proc**(*n*)
local *S*, *i*;
S := 0; **for** *i* **to** *n* – 1 **do if** irem(*n*, *i*) = 0 **then** *S* := *S* + *i* **end if end do**; **return** *S*
end proc

```
> somdivpropres(5);
```

1

```
> somdivpropres(6);
```

6

```
> amiable:=proc(a,b)
  if a<>b and a=somdivpropres(b) and b=somdivpropres(a) then
  return true; else return false;
  end if;
  end proc;
```

amiable := **proc**(*a*, *b*)
if *a* ≠ *b* **and** *a* = somdivpropres(*b*) **and** *b* = somdivpropres(*a*) **then return** *true*
else return *false*
end if
end proc

```
> amiable(220,284);
```

true

```
> amiable(23,18);
```

false

```
> Listeamiables:=proc(N)
  local a,b,L;
  L:=[];
  for b from 1 to N do
  for a from 1 to b-1 do
  if amiable(a,b)=true then L:=[op(L),[a,b]];end if;
  end do;
```

```
  end do;
  return L;
  end proc;
```

Listeamiable := **proc**(*N*)

local *a*, *b*, *L*;

L := [ ];

for *b* **to** *N* **do**

for *a* **to** *b* – 1 **do if** amiable(*a*, *b*) = *true* **then** *L* := [op(*L*), [*a*, *b*]] **end if end do**

end do;

return *L*

end proc

```
> Listeamiables(10);
```

[ ]

```
> Listeamiables(300);
```

[[220, 284]]

```
> #EXO 5 :
> # Version itérative
> entier:=proc(n)
  local i,E;
  E:={};
  if n=0 then return E;
  else for i from 1 to n do
  E:=E union {E};
  end do;
  end if;
  return E;
  end proc;
```

entier := **proc**(*n*)

local *i*, *E*;

E := { };

if *n* = 0 **then return** *E* **else for** *i* **to** *n* **do** *E* := *E* union { *E* } **end do end if**;

return *E*

end proc

```
> entier(0);
```

{ }

```
> entier(1);
```

{{ }}

```
> entier(2);
```

{{ }, {{ }}}

```
> # Version recursive
> entierec:=proc(n)
  if n=0 then return {};
  else return entierec(n-1) union {entierec(n-1)}; end if;
```

```
   end proc;
```

entierec := **proc**(*n*)

if *n* = 0 **then return** { } **else return** entierec(*n* – 1) union { entierec(*n* – 1) } **end if**

end proc

```
> entierec(0);
```

{ }

```
> entierec(1);
```

{ { } }

```
> entierec(2);
```

{ { }, { { } } }

```
> #EXO 6 :
> base10:=proc(n)
   local q,r,L;
   L:=[];
   q:=n;
   while(q<>0) do
   r:=irem(q,10);
   q:=iquo(q,10);
   L:=[r,op(L)];
   end do;
   return L;
   end proc;
```

base10 := **proc**(*n*)

local *q*, *r*, *L*;

L := [ ];

q := *n*;

while *q* ≠ 0 **do** *r* := irem(*q*, 10); *q* := iquo(*q*, 10); *L* := [*r*, op(*L*)] **end do**;

return *L*

end proc

```
> base10(1993);
```

[1, 9, 9, 3]

```
> palindrome:=proc(n)
   local i,L,l,res;
   L:=base10(n);
   l:=nops(L);
   res:=true;
   for i from 1 to l do
   if L[i]<>L[l-i+1] then return false; end if;
   end do;
   return res;
   end proc;
```

palindrome := **proc**(*n*)

local *i*, *L*, *l*, *res*;

```
    L := base10(n);
    l := nops(L);
    res := true;
    for i to l do if L[i] ≠ L[l − i + 1] then return false end if end do;
    return res
end proc
> palindrome(5225);
                                                    true
> palindrome(5223225);
                                                    true
> palindrome(34);
                                                    false
>
```