```
> #TP 3 (MMI1)
> #Exo 1 : Babyloniens
> baby := proc(a,n)
  local i,u;
  u:=1;
  if (n=0) then return u;
  else
  for i from 1 to n do
  u:=1/2*(u+a/u);
  end do;
  end if;
  return u;
  end proc;
```

```
baby := proc(a, n)                                                    (1)
    local i, u;
    u := 1;
    if n = 0 then
        return u
    else
        for i to n do
            u := 1/2 * u + 1/2 * a / u
        end do
    end if;
    return u
end proc
```

```
> evalf(baby(2,2),30);
                    1.41666666666666666666666666667                   (2)
> evalf(baby(2,3),30);
                    1.41421568627450980392156862745                   (3)
> evalf(baby(2,4),30);
                    1.41421356237468991062629557889                   (4)
> evalf(baby(2,5),30);
                    1.41421356237309504880168962350                   (5)
> evalf(baby(2,10),30);
                    1.41421356237309504880168872421                   (6)
> evalf(sqrt(2),30);
                    1.41421356237309504880168872421                   (7)
> #Exo 2 : Tchebychev
> tcheb := proc(n)
  local i,A,B,C;
  A:=1;
  B:=x;
  if n=0 then return A;
  else if n=1 then return B;
  else
  for i from 2 to n do
  C:=2*x*B - A;
  A:=B;
  B:=C;
  end do;
  end if;
  end if;
  return C;
  end proc;
>
>
>
```

```
tcheb := proc(n)                                                        (8)
     local i, A, B, C;
     A := 1;
     B := x;
     if n = 0 then
          return A
     else
          if n = 1 then
               return B
          else
               for i from 2 to n do
                    C := 2 * x * B − A;
                    A := B;
                    B := C
               end do
          end if
     end if;
     return C
end proc
```

```
> tcheb(0), tcheb(1), tcheb(2), tcheb(10);
```

$$\begin{aligned}
&1, x, 2x^2-1, 2x(2x(2x(2x(2x(2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x^2-1) \qquad (9)\\
&\quad +x)-2x(2x(2x^2-1)-x)+2x^2-1)-2x(2x(2x(2x^2-1)-x)-2x^2+1)\\
&\quad +2x(2x^2-1)-x)-2x(2x(2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x^2-1)+x)\\
&\quad +2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x(2x(2x(2x(2x^2-1)-x)-2x^2\\
&\quad +1)-2x(2x^2-1)+x)-2x(2x(2x^2-1)-x)+2x^2-1)\\
&\quad +2x(2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x^2-1)\\
&\quad +x)-2x(2x(2x(2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x^2-1)\\
&\quad +x)-2x(2x(2x^2-1)-x)+2x^2-1)-2x(2x(2x(2x^2-1)-x)-2x^2+1)\\
&\quad +2x(2x^2-1)-x)+2x(2x(2x(2x(2x^2-1)-x)-2x^2+1)-2x(2x^2-1)\\
&\quad +x)-2x(2x(2x^2-1)-x)+2x^2-1
\end{aligned}$$

```
> sort(expand (tcheb(10)));
```

$$512\,x^{10}-1280\,x^{8}+1120\,x^{6}-400\,x^{4}+50\,x^{2}-1 \qquad (10)$$

```
> eval(%,x=cos(a));
```

$$512\cos(a)^{10}-1280\cos(a)^{8}+1120\cos(a)^{6}-400\cos(a)^{4}+50\cos(a)^{2}-1 \qquad (11)$$

```
> combine(%,trig);
```

$$\cos(10\,a) \qquad (12)$$

```
> #Exo 3 :
> facto:=proc(n)
  local i,s;
  s:=1;
  if n=0 then return s; else
  for i from 1 to n do
  s:=s*i;
  end do;
  end if;
  return s;
  end proc;
>
```

```
facto := proc(n)                                                    (13)
    local i, s;
    s := 1;
    if n = 0 then
        return s
    else
        for i to n do
            s := s * i
            end do
        end if;
        return s
    end proc
> facto(0);
                                        1                                    (14)
> facto(1);
                                        1                                    (15)
> facto(2);
                                        2                                    (16)
> facto(3);
                                        6                                    (17)
> facto(4);
                                        24                                   (18)
> facto(5),5!;
                                    120, 120                                 (19)
> factorec:=proc(n)
  if (n=0) then return 1;
  else return n*factorec(n-1);
  end if;
  end proc;
  factorec := proc(n) if n = 0 then return 1 else return n * factorec(n − 1) end if end proc    (20)
> factorec(0);
                                        1                                    (21)
> factorec(1);
                                        1                                    (22)
> factorec(2);
                                        2                                    (23)
> factorec(3);
                                        6                                    (24)
> factorec(4);
                                        24                                   (25)
> factorec(5);
                                       120                                   (26)
> # Exo 4 :
> fibo:=proc(n)
  local i,a,b,c;
```

```
a:=0;
b:=1;
if n=0 then return a; else if n=1 then return b; else
for i from 2 to n do
s:=a+b;
a:=b;
b:=s;
end do;
end if;
end if;
return s;
end proc;
Warning, `s` is implicitly declared local to procedure `fibo`                (27)
```

```
fibo := proc(n)
    local i, a, b, c, s;
    a := 0;
    b := 1;
    if n = 0 then
        return a
    else
        if n = 1 then
            return b
        else
            for i from 2 to n do
                s := a + b;
                a := b;
                b := s
            end do
        end if
    end if;
    return s
end proc
```

```
> fibo(0);
                                  0                                  (28)
```

```
> fibo(1);
                                  1                                  (29)
```

```
> fibo(2);
                                                                     (30)
```

```
                                   1                              (30)
> fibo(3);
                                   2                              (31)
> fibo(4);
                                   3                              (32)
> fibo(5);
                                   5                              (33)
> fibo(6);
                                   8                              (34)
> fibo(7);
                                  13                              (35)
> fibo(8);
                                  21                              (36)
> fibo(9);
                                  34                              (37)
> fibo(10);
                                  55                              (38)
> fiborec := proc(n)
  if n=0 then return 0; else if n=1 then return 1; else
  return fiborec(n-1)+fiborec(n-2);
  end if; end if;
  end proc;
fiborec := proc(n)                                                (39)
    if n = 0 then
        return 0
    else
        if n = 1 then
            return 1
        else
            return fiborec(n - 1) + fiborec(n - 2)
        end if
    end if
end proc
> fiborec(0);
                                   0                              (40)
> fiborec(1);
                                   1                              (41)
```

```
> fiborec(2);
                                  1                                  (42)
> fiborec(3);
                                  2                                  (43)
> fiborec(4);
                                  3                                  (44)
> fiborec(5);
                                  5                                  (45)
> fiborec(6);
                                  8                                  (46)
> fiborec(7);
                                 13                                  (47)
> fiborec(8);
                                 21                                  (48)
> fiborec(9);
                                 34                                  (49)
> fiborec(10);
                                 55                                  (50)
> #Exo 5 :
> restart;
> recurrente:=proc(f,u0,n)
  local i, u,t;
  t:=u0;
  if(n=0) then return t;
  else
  for i from 1 to n do
  u:=f(t);
  t:=u;
  end do;
  end if;
  return u;
  end proc;
```

```
recurrente := proc(f, u0, n)                                         (51)
   local i, u, t;
   t := u0;
   if n = 0 then
       return t
   else
```

```
          for i to n do
               u := f(t);
               t := u
          end do
     end if;
     return u
end proc
```

```
> recurrente2 := proc(f,u0,n)
  if (n=0) then return u0; else
  return f(recurrente2(f,u0,n-1));
  end if;
  end proc;
```

```
recurrente2 := proc(f, u0, n)                                                    (52)
     if n = 0 then
          return u0
     else
          return f(recurrente2(f, u0, n − 1))
     end if
end proc
```

```
> f:=x->2*x;
```

$$f := x \rightarrow 2\,x \tag{53}$$

```
> recurrente(f,1,5),recurrente2(f,1,5);
```

$$32,\ 32 \tag{54}$$

```
> #Exerice 6 :
  feigenbaum := proc(a,u0,n)
  local i, u,t;
  t:=u0;
  if (n=0) then return t;
  else
  for i from 1 to n do
  u:=a*t*(1-t);
  t:=u;
  end do;
  end if;
  return u;
  end proc;
```

```
feigenbaum := proc(a, u0, n)                                                     (55)
     local i, u, t;
     t := u0;
```

```
    if n = 0 then
        return t
    else
        for i to n do
            u := a * t * (1 − t);
            t := u
        end do
    end if;
    return u
end proc
```

```
> feigenbaum2 := proc(a,u0,n)
  if n=0 then return u0; else
  return a*feigenbaum2(a,u0,n-1)*(1-feigenbaum2(a,u0,n-1));end if;
  end proc;
```

```
feigenbaum2 := proc(a, u0, n)                                                    (56)
    if n = 0 then
        return u0
    else
        return a * feigenbaum2(a, u0, n − 1) * (1 − feigenbaum2(a, u0, n − 1))
    end if
end proc
```

```
> feigenbaum(1.5,0.5,5), feigenbaum2(1.5,0.5,5);
                         0.3354052689, 0.3354052689                               (57)
```

```
> feigenbaum(1.5,0.5,10), feigenbaum2(1.5,0.5,10);
                         0.3333973076, 0.3333973076                               (58)
```

```
> feigenbaum(1.5,0.5,15), feigenbaum2(1.5,0.5,15);
                         0.3333353318, 0.3333353318                               (59)
```

```
> #Exo 7 :
```

```
> addrec:=proc(m,n)
  if(n=0) then return m;
  else return addrec(m,n-1)+1;
  end if;
  end proc;
 addrec := proc(m, n) if n = 0 then return m else return addrec(m, n − 1) + 1 end if end proc   (60)
```

```
> addrec(2,2);
                                      4                                           (61)
```

```
> addrec(3,0);
                                                                                  (62)
```

```
                                   3                                   (62)
> addrec(3,5);
                                   8                                   (63)
> addrec(3,1);
                                   4                                   (64)
> multrec:=proc(m,n)
  if (n=0) then return 0;
  else return multrec(m,n-1)+m;
  end if;
  end proc;
 multrec := proc (m, n) if n = 0 then return 0 else return multrec(m, n − 1) + m end if end proc  (65)
> multrec(3,2);
                                   6                                   (66)
> multrec(3,0);
                                   0                                   (67)
> multrec(3,1);
                                   3                                   (68)
> multrec(3,5);
                                  15                                   (69)
> exprec:=proc(m,n)
  if (n=0) then return 1;
  else return exprec(m,n-1)*m;
  end if;
  end proc;
   exprec := proc (m, n) if n = 0 then return 1 else return exprec(m, n − 1) * m end if end proc  (70)
> exprec(0,0);
                                   1                                   (71)
> exprec(2,0);
                                   1                                   (72)
> exprec(3,1);
                                   3                                   (73)
> exprec(3,2);
                                   9                                   (74)
> exprec(3,3);
                                  27                                   (75)
> #Exo 8 :
> dichotomie:=proc(f,a,b,epsilon)
  local o,e,d,l;
```

```
o:=a;
e:=b;
d:=evalf((a+b)/2);
l:=evalf(abs(b-a));
if epsilon > l then return d; else
while (l>=epsilon) do
if f(o)*f(d)>0 then o:=d;d:=evalf((o+e)/2);l:=evalf(abs(e-o));
else e:=d;d:=evalf((o+e)/2);l:=evalf(abs(e-o));
end if;
end do;
end if;
return d;
end proc;
```

```
dichotomie := proc(f, a, b, ε)                                        (76)
     local o, e, d, l;
     o := a;
     e := b;
     d := evalf(1/2*a + 1/2*b);
     l := evalf(abs(b − a));
     if l < ε then
         return d
     else
         while ε <= l do
             if 0 < f(o)*f(d) then
                 o := d;
                 d := evalf(1/2*o + 1/2*e);
                 l := evalf(abs(e − o))
             else
                 e := d;
                 d := evalf(1/2*o + 1/2*e);
                 l := evalf(abs(e − o))
             end if
         end do
     end if;
     return d
end proc
```

```
> dichotomie(sin,-Pi/2,Pi/2,0.0001);
```

(77)

$$-0.00004793689962 \tag{77}$$

```
> dichotomie(sin,-Pi/2,Pi/2,0.00000001);
```

$$-2.925836155\ 10^{-9} \tag{78}$$

```
> g:=x->x^5+3*x-7;
```

$$g := x \rightarrow x^5 + 3\,x - 7 \tag{79}$$

```
> dichotomie(g,0,2,0.001);
```

$$1.263183594 \tag{80}$$

```
> fsolve(g(x)=0);
```

$$1.262822860 \tag{81}$$

```
>
```