```
> #1
> reduction := proc (L)
  local i,longueur, Ltemp;
  Ltemp:=[];
  if nops(L)=0 or nops(L)=1 then return L;
  else
  longueur:=nops(L);
  for i from 1 to nops(L) do
  if i<=nops(L)-1 and L[i]=-L[i+1]  then
  i:=i+1;
  else Ltemp:=[op(Ltemp),L[i]];
  end if;
  end do;
  end if;
  return Ltemp;
  end proc;
```

reduction := **proc**(*L*)
local *i*, *longueur*, *Ltemp*;
Ltemp := [ ];
if nops(*L*) = 0 **or** nops(*L*) = 1 **then return** *L*
else
longueur := nops(*L*);
for *i* **to** nops(*L*) **do**
if *i* ≤ nops(*L*) − 1 **and** *L*[*i*] = –*L*[*i* + 1] **then** *i* := *i* + 1
else *Ltemp* := [op(*Ltemp*), *L*[*i*]]
end if
end do
end if;
return *Ltemp*
end proc

```
> reduction([]);
```

[ ]

```
> reduction([-a]);
```

[–*a*]

```
> reduction([-a,a]);
```

[ ]

```
> reduction([-a,a,b,c,-c,-c,-d,d,a,b]);
```

[*b*, –*c*, *a*, *b*]

```
> reduction([-a,a,b,c,-c,c,-c,-c,c,-c,c,-d,d,a,-b,b]);
```

[*b*, *a*]

```
> reduction([-a,b,-b,a]);
```

[–*a*, *a*]

```
> #On voit sur le dernier exemple qu'avec la procédure "reduction"
  telle qu'elle est implémentée, on ne réduit pas les
  #nouveaux couples [-x,x] ou [x,-x] créés lors d'une réduction
  antérieure. Il faut donc corriger cela. On reprend cette
  #procédure et on rajoute un indicateur "red" qui nous dit si une
  réduction a été effectuée (red=true) ou non #(red=false).
```

```
>
  reduction2 := proc (L)
  local i,red,longueur, Ltemp;
  Ltemp:=[];
  red:=false;
  if nops(L)=0 or nops(L)=1 then return L,false;
  else
  longueur:=nops(L);
  for i from 1 to nops(L) do
  if i<=nops(L)-1 and L[i]=-L[i+1]  then
  i:=i+1;
  red:=true;
  else Ltemp:=[op(Ltemp),L[i]];
  end if;
  end do;
  end if;
  return Ltemp,red;
  end proc;
```

reduction2 := **proc**(*L*)
local *i*, *red*, *longueur*, *Ltemp*;
  Ltemp := [ ];
  red := *false*;
  if nops(*L*) = 0 **or** nops(*L*) = 1 **then return** *L*, *false*
  else
    longueur := nops(*L*);
    for *i* **to** nops(*L*) **do**
      if *i* ≤ nops(*L*) – 1 **and** *L*[*i*] = –*L*[*i* + 1] **then** *i* := *i* + 1; *red* := *true*
      else *Ltemp* := [op(*Ltemp*), *L*[*i*]]
      end if
    end do
  end if;
  return *Ltemp*, *red*
end proc

```
> reduction2([-a,a,b,c,-c,c,-c,-c,c,-c,c,-d,d,a,-b,b]);
```

[*b*, *a*], *true*

```
> reduction2([-a,b,-b,a]);
```

$[-a, a], true$

```
> reduction2([-a,b,b,a]);
```

$[-a, b, b, a], false$

```
> #La procédure "red" fonctionne comme suit : on commence par
  appliquer une fois la procédure reduction2 puis si (au moins)
  #une réduction a été effectuée, on recommence avec la liste
  réduite, et cela tant qu'une réduction a été faite (ce qui
  #pourrait avoir créé une nouvelle réduction).

  red:=proc(L)
  local S;
  S:=NULL;
  S:=reduction2(L);
  while S[2]=true do
  S:=reduction2(S[1]);
  end do;
  return S[1];
  end proc;
>
```

red := **proc**(*L*)
local *S*;
S := *NULL*;
S := reduction2(*L*);
while *S*[2] = *true* **do** *S* := reduction2(*S*[1]) **end do**;
return *S*[1]
end proc

```
> red([-a,a,b,c,-c,c,-c,-c,c,-c,c,-d,d,a,-b,b]);
```

$[b, a]$

```
> red([-a,b,-b,a]);
```

$[\ ]$

```
> S:=reduction2([-a,b,-b,a]);
```

$S := [-a, a], true$

```
> S[1];
```

$[-a, a]$

```
> S:=reduction2(S[1]);
```

$S := [\ ], true$

```
> S:=reduction2(S[1]);
```

$S := [\ ], false$

```
> #Question 2 (voir la correction du contrôle du 20-11-2009).

> #Question 3

> produit2:=proc(L,M)
  return red([op(L),op(M)]);
```

```
  end proc;
```

produit2 := **proc**(*L*, *M*) **return** red([op(*L*), op(*M*)]) **end proc**

```
> produit2([-a,b,-b,c,a],[-a,-c,-b,b,c,-c,a]);
```

[ ]

```
> produit2([-a,b,-b,-c,a],[-a,-c,-b,b,c,-c,a]);
```

[–*a*, –*c*, –*c*, *a*]

```
> #Question 4 :
> inverse:=proc(L)
  local i,Ltemp,Inv;
  Ltemp:=red(L);
  Inv:=[];
  if nops(Ltemp)=0 then return [];
  else
  for i from 1 to nops(Ltemp) do
  Inv:=[-Ltemp[i],op(Inv)];
  end do;
  end if;
  return Inv;
  end proc;
```

inverse := **proc**(*L*)
local *i*, *Ltemp*, *Inv*;
Ltemp := red(*L*);
Inv := [ ];
if nops(*Ltemp*) = 0 **then return** [ ]
else for *i* **to** nops(*Ltemp*) **do** *Inv* := [–*Ltemp*[*i*], op(*Inv*)] **end do**
end if;
return *Inv*
end proc

```
> inverse([-a,-c,c,-a,a,b,d,e,-e,-d,-b,-a,c]);
```

[–*c*, *a*, *a*]

```
> #Question 5 :
  puissance:=proc(n,L)
  local i;
  if n=0 then return []; end if;
  if n=1 then return red(L); end if;
  if n>1 then
  return produit2(L,puissance(n-1,L));
  end if;
  end proc;
```

puissance := **proc**(*n*, *L*)
local *i*;
if *n* = 0 **then return** [ ] **end if**;
if *n* = 1 **then return** red(*L*) **end if**;

```
    if 1 < n then return produit2(L, puissance(n − 1, L)) end if
end proc
> puissance(0,[-a,b,-b,a,c,d]);
                                        [ ]
> puissance(1,[-a,b,-b,a,c,d]);
                                      [c, d]
> puissance(2,[-a,b,-b,a,c,d]);
                                   [c, d, c, d]
> puissance(10,[-a,b,-b,a,c,d]);
        [c, d, c, d, c, d, c, d, c, d, c, d, c, d, c, d, c, d, c, d]
> puissance(2,[-a,b,a]);
                                    [−a, b, b, a]
> puissance(3,[-a,b,a]);
                                 [−a, b, b, b, a]
>
```