```
> produit:=proc(A,B)
  #Variables locales : res pour mettre le résultat, indexA et
  indiceB sont respectivement le premier indice de la liste A
  #(de la droite vers la gauche) et le premier indice de la liste
  B (de la gauche vers la droite) tels que #A[indexA]<>-B[indexB].
  On ne peut donc pas "éliminer" A[indexA] et B[indexB].
  #La variable i est un compteur.

  local res, indexA, indexB,i;
  res:=[];

  #1er cas : A est [], alors on renvoie B et si B=[], alors on
  renvoie A.
  if nops(A)=0 then return B; else if nops(B)=0 then return A; end
  if; end if;

  #2nd cas : A et B sont non vides. On doit donc calculer indexA
  et indexB. On initialise indexA à nops(A) et indexB à 1.
  if (nops(A)>0 and nops(B)>0) then
  indexA:=nops(A);
  indexB:=1;

  #Tant que indexA>0, indexB<=nops(B) et A[indexA]=-B[indexB], on
  met à jour indexA et indexB. Ainsi indexA:=indexA-1 et
  #indexB:=indexB+1.

  while ((indexA>0 and (nops(B)-indexB)>=0)) and
  (A[indexA]=-B[indexB] or B[indexB]=-A[indexA]) do
  indexA:=indexA-1;
  indexB:=indexB+1;
  end do;
  end if;

  #Maintenant que l'on a calculé indexA et indexB, on construit le
  résultat.

  #Si indexA=0, on a éliminé tous les éléments de A. Si
  indexB<=nops(B), le résultat res est la liste B de i=indexB
  #jusqu'à la fin de B.

  if (indexA=0) then
      if (indexB<=nops(B)) then
         for i from indexB to nops(B) do
           res:=[op(res),B[i]];
         end do;
         return res;
```

```
  #Sinon (indexB>nops(B)), on renvouie la liste vide car on a
  éliminé tous les éléments de chacune des deux listes.
      else return [];
      end if;

  #Sinon (indexA>0), on construit le résultat res. Si
  indexB=nops(B)+1, on a éliminé tous les éléments de B, et le
  résultat #res est construit comme la liste A de i=1 à indexA.

  else
      if (indexB=nops(B)+1) then
         for i from 1 to indexA do
           res:=[op(res),A[i]];
         end do;
         return (res);

  #Sinon (indexB<=nops(B)), on construit le résultat res comme la
  concaténation de A et B à laquelle on retire tous les
  #éléments qui se sont éliminés deux à deux entre A et B,
  c'est-à-dire les éléments de A de indexA+1 à nops(A) et ceux de
  #B de 1 à indexB-1. Il ne reste donc dans res que les éléments
  de A de 1 à indexA puis ceux de B de indexB à nops(B).
     else
         for i from 1 to indexA do
           res:=[op(res),A[i]]; end do;
         for i from indexB to nops(B) do
           res:=[op(res),B[i]]; end do;
         return res;
     end if;
  end if;
  end proc;
```

produit := **proc**(*A*, *B*)
local *res*, *indexA*, *indexB*, *i*;
res := [ ];
if nops(*A*) = 0 **then return** *B* **else if** nops(*B*) = 0 **then return** *A* **end if end if**;
if 0 < nops(*A*) **and** 0 < nops(*B*) **then**
indexA := nops(*A*);
indexB := 1;
while 0 < *indexA* **and** 0 ≤ nops(*B*) − *indexB* **and**
(*A*[*indexA*] = −*B*[*indexB*] **or** *B*[*indexB*] = −*A*[*indexA*]) **do**
indexA := *indexA* − 1; *indexB* := *indexB* + 1
end do

```
    end if;
    if indexA = 0 then
        if indexB ≤ nops(B) then
            for i from indexB to nops(B) do res := [op(res), B[i]] end do; return res
        else return [ ]
        end if
    else
        if indexB = nops(B) + 1 then
            for i to indexA do res := [op(res), A[i]] end do; return res
        else
            for i to indexA do res := [op(res), A[i]] end do;
            for i from indexB to nops(B) do res := [op(res), B[i]] end do;
            return res
        end if
    end if
end proc
> produit([],[]);produit([-a],[-a]);produit([-a],[a]);produit([],[
  x,y,-z]);
                                    [ ]
                                 [-a, -a]
                                    [ ]
                                 [x, y, -z]
> produit([x,y,-z],[]);
                                 [x, y, -z]
> produit([-c,x,a,b,x,y,-z],[z,-y,-x,-b,-a,c]);
                                  [-c, x, c]
> produit([-c,x,a,b,x,y,-z],[z,-y,-x,-b,-a,-x,c]);
                                    [ ]
>
```