3/2-approximation algorithm of Ailon to compute un consensus of "ens", where each ranking in "ens" is a ranking of [n]. The return consensus is always a full-ranking

```
Ailon3Demi:=proc(ens,n)
     local valeurs,tabX,i,tri;
     valeurs:={seq(i,i=1..n)};
     tabX:=genereX(ens,n);
     tri:=LPkWikSort(valeurs,tabX);
tri;
end:
```

Compute by linear programming the x's needed by algorithm LPkWikSort

```
genereX:=proc(ens,n)
     local eq,contr,matW,matX,listeVar,listeVal,rep;
     matW:=calculW(ens,n);
     contr:=genereContraintes(n);
     eq:=EquationMinimiser(matW,n);
     rep:=LPSolve(eq,contr);
     listeVar:=[op(indets(rep))];
     listeVal:=subs(op(2,rep),listeVar);
     matX:=tableX(listeVar,listeVal,n);
matX;
end:
```

Compute the w[i][j] values (cost of pair [i][j] with respect to set "ens") of AIlon

```
calculW:=proc(ens,n)
   local m,i,j,k,W,pos;
   m:=nops(ens);
   W:=Array(1..n,1..n);
   for k from 1 to m do
        pos:=position(ens[k],n);
        for i from 1 to n do
           for j from 1 to n do
                if pos[i] < pos[j] then
                      W[i,j]:=W[i,j]+(1/m);
                fi;
           od;
       od;
od;
W;
end:
```

Compute the equation to minimize by LP

```
EquationMinimiser:=proc(matriceW,n)
```

```
    local i,j,somme;
    somme:=0;
    for i from 1 to n-1 do
        for j from i+1 to n do
              somme:=somme+(x[i,j]*matriceW[j,i])+(x[j,i]*matriceW[i,j]);
      od;
    od;
somme;
end:
```

Compute the constraints for the LP minimization

```
genereContraintes:=proc(n)
      local i,j,k,contr;
      contr:={};
      for i from 1 to n-1 do
          for j from i+1 to n do
              contr:=contr union {x[i,j] >=0,x[j,i]>=0,x[i,j]+x[j,i]=1}
          od;
      od;
      for i from 1 to n do
          for j from 1 to n do
              for k from 1 to n do
                      if evalb(i <> j) and evalb(i <> k) and evalb (j<>k) then
                          contr:=contr union {x[i,j] <= x[i,k] + x[k,j]};
                    fi;
            od;
        od;
od;
contr;
end:
```

Algorithm LPKWikSort of Ailon taking as input the domain V of the set of rankings and a matrix x of values x[i][j], for i,j in [n]. These x[i][j] values are compute with an LP algorithm.

```
LPkWikSort:=proc(V,tableX)
      local rep,L,R,al,roll,pourcent,pivot,i,fh,per;
      if V = {} then
          rep:=[];
      else
          L:={};
          R:={};
          roll:=rand(1..nops(V));
          al:=roll();
```

```
     pivot:=V[al];
     for i from 1 to nops(V) do
       if V[i] <> pivot then
         fh:=100*convert(fonctionH(tableX[V[i],pivot]),float,2);
         if evalb(fh = 0) then
            R:=R union {V[i]};
         elif evalb (fh = 1) then
            L:=L union {V[i]};
         else
           pourcent:=rand(1..100);
           per:=pourcent();
           if evalb(per <= fh) then
               L:=L union {V[i]};
           else
               R:=R union {V[i]};
           fi;
        fi;
       fi;
    od;
    fi;
  rep:=[op(LPkWikSort(L,tableX)),{pivot},op(LPkWikSort(R,tableX)];
rep;
end:
```

Compute the value of function h of Ailon on a certain x

```
fonctionH:=proc(x)
   local rep;
   if x >= 0 and x <=1/6 then
      rep:=0;
   elif x > 1/6 and x <=5/6 then
      rep:=(3/2*x) - (1/4);
   else
      rep:=1;
   fi;
rep;
end:
```