Corrigé

1. a) Algorithme 3.2.2, p. 8

b)

i	liste.items					n_aff	n_comp
	M	O	I	E	N		
1	M	O	I	E	N	1	1
2	I	M	O	E	N	4	2
3	E	I	M	O	N	5	3
4	E	I	M	N	O	3	2

nombre d'affectations de strings : 1 + 4 + 5 + 3 = 13

nombre de comparaisons de strings : 1 + 2 + 3 + 2 = 8

2 a) f('Moien')

=> 'n'+f('Moie')+'n'

=> 'n'+'e'+f('Moi')+'e'+'n'

=> 'ne'+'i'+f('Mo')+'i'+'en'

=> 'nei'+'o'+f('M')+'o'+'ien'

=> 'neio'+'M'+'oien' [length(x)<=1]

=> 'neioMoien'

b) La fonction renvoie comme string-résultat le string-argument *x* renversé, suivi de *x* non renversé, sans qu'il y ait répétition de la première lettre de *x* au milieu du string-résultat.

c) voir la fonction f_iter dans le listing annexé

d) Si la longueur du string-argument est inférieure ou égale à 2, g et f produisent le même résultat. Par contre, si la longueur du string-argument est strictement supérieure à 2, g ne produit pas le même résultat que f : pour tout i, la concaténation x := x[i] + x ajoute toujours la **même** lettre à gauche du string x, à savoir la 2^{e} lettre de l'argument initial x.

```
program exercice2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

function f(x:string):string;
begin
  if length(x)<=1 then f:=x
  else f:=x[length(x)]+f(copy(x,1,length(x)-1))+x[length(x)]
end;

function f_iter(x:string):string;
var i:integer;
    s:string;
begin
  s:=x;
  for i:=2 to length(x) do
    s:=x[i]+s;
  f_iter:=s
end;

function g(x:string):string;
var i:integer;
begin
  for i:=2 to length(x) do
    x:=x[i]+x;
  g:=x
end;

begin
  writeln(f('Moien'));
  writeln(f('a'));          (* quelques tests *)
  writeln(f('ab'));
  writeln(f('abc'));
  writeln(f('abcd'));
  writeln(f_iter('Moien'));
  writeln(f_iter('a'));
  writeln(f_iter('ab'));
  writeln(f_iter('abc'));
  writeln(f_iter('abcd'));
  writeln(g('Moien'));
  writeln(g('a'));
  writeln(g('ab'));
  writeln(g('abc'));
  writeln(g('abcd'));
  readln
end.
```