Encore un petit test [*] [†] : Correction

Exercice 1 : programme mystère.

On considère le programme Ada suivant :

```
(1) Procedure Keskessafai is
(2)   n : integer := 7 ;
(3)   t : array (1 .. n) of integer range 0 .. 20 ;
(4)   r : integer := 0;
(5) begin
(6)   t := (12,3,14,5,10,19,9) ;
(7)   for i in 1 .. n loop
(8)     if (i = 1) then Put("[ ") ; end if ;
(9)     Put(t(i),0) ;
(10)    if (i = n) then Put(" ]") ;
(11)    else Put("; ") ;
(12)    end if ;
(13)  end loop ;
(14)  for i in t'range loop
(15)    if (t(i) >= 10) then
(16)      r := r + 1 ;
(17)    end if ;
(18)  end loop ;
(19)  New_Line ;
(20)  Put("Le résultat est : ") ;
(21)  Put(r,0) ;
(22) end Keskessafai ;
```

(Q.1) Quel est l'affichage produit par le programme `Keskessafai` ?

[12 ; 3 ; 14 ; 5 ; 10 ; 19 ; 9]
Le résultat est : 4

(Q.2) Que se passe-t-il si on remplace la ligne `(6)` par `t:= (0;0;0;0;0;0;42)` ? Expliquer Il y aura une erreur lors de l'exécution, car par définition `t` ne peut contenir que valeurs entières comprises entre 0 et 20. Et très formellement, il y aura aussi une erreur de syntaxe (les points-virgules doivent être remplacés par des virgules).

*. C'est le dernier.
†. Ou pas.

Et si on la remplace par `t:=(10,29)` ? Expliquer Il y aura une erreur lors de l'exécution, car par définition `t` est un tableau de taille 7 et doit donc contenir 7 valeurs lors d'une affectation globale. Il y aura aussi la même erreur que la question précédente (car 29 n'est pas dans la plage de valeurs définie).

(Q.3) On veut remplacer la ligne `(3)` par `t : typeTab1(1 .. n) ;`
Donner l'instruction pour définir le type `typeTab1` :

```
type typeTab1 is array (integer range <>) of integer range 0 .. 20 ;
```

(Q.4) On veut remplacer la ligne `(3)` par `t : typeTab2 ;`
Donner l'instruction pour définir le type `typeTab2` :

```
type typeTab2 is array (1 .. n) of integer range 0 .. 20 ;
```

(Q.5) À quoi sert la première boucle du programme `Keskessafai` ? Elle sert à afficher les valeurs du tableau, sous la forme `[ valeur1 ; valeur2 ; ... ; valeur7 ]`

(Q.6) À quoi sert la deuxième boucle du programme `Keskessafai` ? Elle sert à compter le nombre d'éléments du tableau dont la valeur est supérieure ou égale à 10

Exercice 2 : définitions de tableaux.

1. Donner les instructions pour définir un type `tabBool` de tableaux de taille 5 contenant des booléens

   ```
   type tabBool is array (1..5) of boolean ;
   ```

2. Donner les instructions pour déclarer une variable `t` de type `tabBool` et affecter la valeur `true` à toutes les cases

   ```
   t : tabBool := (others => true) ;
   ```

   Solution avec une boucle (plus lourde) :

   ```
   t : tabBool ; -- déclaration de la variable
   for i in t'range loop -- boucle pour affecter toutes les cases à true
     t(i) := true ;
   end loop ;
   ```

3. Donner les instructions pour déclarer une variable `t` de type `tabBool` et affecter la valeur `true` aux 3 premières cases et `false` aux 2 dernières

   Plusieurs solutions possibles :

   ```
   t : tabBool := (1..3 => true, 4..5 => false) ; (on peut remplacer 4..5 par 4|5)
   t : tabBool := (true,true,true,false,false) ;
   t : tabBool := (true,true,true,others=>false) ;
   ```

   Solution avec une boucle :

   ```
   for i in t'range loop
     if (i <= 3)
       then t(i) := true ;
       else t(i) := false ;
     end if ;
   end loop ;
   ```

Exercice 3 : lancers de dés.

1. Écrire un programme `LancersDés` qui, pour un entier n à valeur fixée (lors de la déclaration de la variable `n`), remplit un tableau de taille n avec les résultats de n lancers de dés ‡ (le résultat d'un lancer de dé est une valeur entre 1 et 6).
   Réponse :

```
n : integer := valeur_fixe ;
type Lancers is array (1 .. n) of integer range 1 .. 6 ;
tab : Lancers ;
procedure LancersDés is
  for i in tab'range loop                          -- remplissage du tableau
    tab(i) := Lancer(1,6) ;
  end loop ;
  for i in tab'range loop                            -- affichage du tableau
    Put(tab(i)) ;
    Put(" ; ") ;
  end loop ;
end LancerDés ;
```

2. Compléter le programme précédent pour qu'il compte et affiche le nombre de 6 obtenus dans le tableau des n lancers.
   Réponse :

```
n : integer := valeur_fixe ;
nbre_six : integer := 0 ;
type Lancers is array of (1 .. n) of integer range 1 .. 6 ;
tab : Lancers ;
procedure LancersDés is
  for i in tab'range loop
    tab(i) := Lancer(1,6) ;                        -- remplissage du tableau
    if ( tab(i) = 6 ) then                                 -- comptage des 6
      nbre_six := nbre_six + 1 ;
    end if ;
  end loop ;
  for i in tab'range loop                            -- affichage du tableau
    Put(tab(i)) ;
    Put(" ; ") ;
  end loop ;
  New_Line ;
  Put("Il y a eu " & integer'image(nbre_six) & " six.") ;
end LancerDés ;
```

‡. On suppose que l'on a à disposition une fonction `Lancer(a,b)` qui fournit un nombre entier aléatoire compris entre a et b.

3. Écrire un programme `StatsLancerDés` qui affiche successivement le nombre de 6 obtenus en 10 lancers, puis en 20 lancers, puis en 30 lancers ... et enfin en 100 lancers. On affichera également à chaque fois le pourcentage de 6 obtenus pour chaque paquet de lancers effectués. *Le programme* `StatsLancerDés` *ne devra pas utiliser de tableaux.*
Réponse :

```
nbre_six, pourcent : integer := 0 ;
procedure StatsLancersDés is
  for i in 1 .. 100 loop
    if ( Lancer(1,6) = 6 ) then
      nbre_six := nbre_six + 1 ;
    end if ;
    if (i mod 10 = 0) then
    Put_Line("Il y a eu " & integer'image(nbre_six) & " six sur " &
integer'image(i) & " lancers.") ;
    pourcent := integer( float(nbre_six * 100) / float(i) ) ;
    Put_Line("Cela fait environ" & integer'image(pourcent) & " % de six.") ;
  end loop ;
end StatsLancerDés ;
```