DS 1 d'option informatique - MPSI - Correction

Question 1.a –

```
let rec length (li: 'a list): int = match li with
  | [] -> 0
  | e :: q -> 1 + length q;;
```

Question 1.b –

```
let rec concat (li1: 'a list) (li2: 'a list): 'a list = match li1 with
  | [] -> li2
  | e1 :: q1 -> e1 :: concat q1 li2;;
```

Question 1.c –

```
let rec filter (li: 'a list) (pred: 'a -> bool) = match li with
  | [] -> []
  | e :: q when pred e -> e :: (filter q pred)
  | _ :: q -> filter q pred;;
```

Question 2.a – Soit c_0 un collier. On a :

→ $c_0 = \varepsilon \cdot c_0$, donc ε est un préfixe de c_0.

→ $c_0 = c_0 \cdot \varepsilon$, donc ε est un suffixe de c_0.

Question 2.b – Soit c_0 un collier. On a :

→ $c_0 = c_0 \cdot \varepsilon$, donc c_0 est un préfixe de lui-même.

→ $c_0 = \varepsilon \cdot c_0$, donc c_0 est un suffixe de lui-même.

Question 2.c – On a deux cas :

→ Si c est un préfixe de c_0, alors il existe c_1 tel que $c_0 = c \cdot c_1$. Ainsi, $c_0 = \varepsilon \cdot c \cdot c_1$ et donc c est un facteur de c_0.

→ Si c est un suffixe de c_0, alors il existe c_1 tel que $c_0 = c_1 \cdot c$. Ainsi, $c_0 = c_1 \cdot c \cdot \varepsilon$ et donc c est un facteur de c_0.

Question 3.a – Si on pose $c = 12021$, alors $c_0 = \varepsilon \cdot (c \cdot c) \cdot \varepsilon$. Donc $c \cdot c$ est un facteur de c_0. Ainsi, par définition de "collier solution" :

1202112021 n'est pas un collier solution.

Question 3.b – Pour cela, on prend un collier solution de taille 2 et on y ajoute une perle. La seule contrainte est que les deux dernières perles doivent être différentes. On obtient 12 colliers :

010	012	020	021	101	102
120	121	201	202	210	212

Question 3.c – Le collier c est de taille 6, il faut donc ajouter deux perles à droite de c. Étant donné que la dernière perle de c est un 0, le première perle ajoutée ne peut pas être 0. De plus, les deux perles à ajouter doivent être différentes. On obtient 4 possibilités :

$$c_1 = 10201010 \qquad c_2 = 10201012 \qquad c_3 = 10201020 \qquad c_4 = 10201021$$

On remarque que :

→ c_1 et c_2 ne sont pas des colliers solutions car ils admettent $01 \cdot 01$ comme facteur.

→ c_3 n'est pas un collier solution car il admet $1020 \cdot 1020$ comme facteur.

→ c_4 est bien un collier solution.

En conclusion :

Le seul collier solution de taille 8 dont c est un préfixe est $c_4 = 10201021$.

Question 3.d – Supposons disposer d'un collier c' de taille $n \geqslant 12$ dont c est un suffixe. On a donc $c' = p \cdot c$ où $p \in \{0, 1, 2\}$. Dans les trois cas, c' n'est pas un collier solution :

→ Pour $p = 0$: $c' = 001021201021$ dont $0 \cdot 0$ est facteur.

→ Pour $p = 1$: $c' = 101021201021$ dont $10 \cdot 10$ est facteur.

→ Pour $p = 2$: $c' = 201021201021$ dont $201021 \cdot 201021$ est facteur.

Question 4 –

```
let rec verif012 (li: int list): bool = match li with
  | [] -> true
  | e :: q -> (e = 0 || e = 1 || e = 2) && verif012 q;;
```

Question 5.a –

```
let rec estPref (c1: collier) (c2: collier): bool = match c1, c2 with
  | [], _ -> true
  | e1::q1, [] -> false
  | e1::q1, e2::q2 -> e1 = e2 && estPref q1 q2;;
```

Question 5.b –

```
(* Hypothèse: n >= 0 *)
let rec couper (c: collier) (n: int): collier * collier = match c, n with
  | c, 0 -> [], c
  | [], _ -> failwith "couper: n > len(c)"
  | e::q, n -> let c1, c2 = couper q (n-1) in
               e :: c1, c2;;
```

Question 5.c –

```
(* Hypothèse: n >= 0 *)
let rec doubleFact (c: collier) (len: int) (n: int) = match c with
  | _ when 2*n > len -> false
  | [] -> n = 0
  | e::q -> let c1, c2 = couper c n in
            (estPref c1 c2) || doubleFact q (len-1) n;;
```

Question 6 –

```
let estSol (c: collier) =
  let len = List.length c in
  let n = ref 1 in
  while 2 * !n <= len && not (doubleFact c len !n) do
    incr n
  done;
  2 * !n > len;;
```

Question 7.a –

```
let rec ajout (li: collier list): collier list = match li with
  | [] -> []
  | c :: q -> (0::c) :: (1::c) :: (2::c) :: (ajout q);;
```

Question 7.b –

```
let rec tousCol (n: int) = match n with
  | n when n < 0 -> failwith "tousCol: n < 0"
  | 0 -> [[]]
  | n -> ajout (tousCol (n-1));;
```

Question 7.c –

```
let tousColSol (n: int) =
  let li1 = tousCol n in
  filter li1 estSol;;
```

Question 8 –

```
let motTM (n: int): mot =
  let s = ref "a" in
  let len = ref 1 in
  for i = 1 to n do
    len := 2 * !len;
    let f i = match !s.[i/2], i mod 2 with
      | 'a', 0 -> 'a'
      | 'a', 1 -> 'b'
      | 'b', 0 -> 'b'
      | 'b', 1 -> 'a'
      | _ -> failwith "Le mot n'est pas valide"
    in
    s := String.init !len f;
  done;
  !s;;
```

Question 9 –

« `mot_of_collier [1; 0; 1; 2; 0; 0]` » vaut `"abaababbaa"`

Question 10.a – On le montre par récurrence sur $n \in \mathbb{N}$:

→ Initialisation. Pour $n = 0$: $s_0 = a$ qui commence bien par un a.

→ Hérédité. Soit $n \in \mathbb{N}$. On suppose que s_n commence par un a. Alors s_{n+1} commence par $\phi(a) = ab$. Donc s_{n+1} commence par a.

Question 10.b – Montrons par récurrence sur $n \in \mathbb{N}$ que $\boxed{\text{le nombre de caractères dans } s_n \text{ est } 2^n}$:

→ Initialisation. Pour $n = 0$, le nombre de caractères dans $s_0 = a$ est 1 et $2^0 = 1$.

→ Hérédité. On suppose que le nombre de caractères dans s_n est 2^n. Comme $s_{n+1} = \phi(s_n)$, s_{n+1} contient deux fois plus caractères que s_n, c'est à dire $2 \times 2^n = 2^{n+1}$.

Question 10.c – Pour $n = 0$, le mot de Thue-Morse $s_0 = a$ ne contient pas trois b consécutifs.

Pour $n > 0$, si on découpe s_n par groupes de deux lettres, alors chacun de ces groupes est égal à $\phi(c)$ avec c une lettre de s_{n-1}. On a donc $\phi(c) \in \{ab, ba\}$. Si on considère x, y, z trois lettres consécutives dans s_n, alors x, y ou y, z forment l'un des groupes évoqués plus haut. Ainsi, $xy \in \{ab, ba\}$ ou $yz \in \{ab, ba\}$. Dans tous les cas, $x = a$ ou $y = a$ ou $z = a$.

Question 10.d – On montre l'existence, puis l'unicité.

Existence. Soit $n \in \mathbb{N}$. À l'aide des questions 10.a et 10.c, on remarque que le mot de Thue-Morse s_n peut être décomposé sous la forme :

$$s_n = t_1 \cdot t_2 \cdot \ldots \cdot t_k \qquad \text{où} \qquad k \in \mathbb{N}^* \text{ et } \forall i \in [\![1, k]\!] : t_k \in \{a, ab, abb\}$$

Ainsi, si on note c le collier défini par :

$$c = p_1 \cdot p_2 \cdot \ldots \cdot p_k \qquad \text{où} \qquad \forall i \in [\![1, k]\!] : p_i = \begin{cases} 0 \text{ si } t_i = a \\ 1 \text{ si } t_i = ab \\ 2 \text{ si } t_i = abb \end{cases}$$

alors « `mot_of_collier c` » s'évalue en s_n.

Unicité. Soient c_1, c_2 deux colliers et t_1, t_2 leurs images par `mot_of_collier` (t_1 et t_2 sont des mots). Pour montrer l'unicité, on doit montrer que si $t_1 = t_2$, alors $c_1 = c_2$. Supposons $t_1 = t_2$ et supposons par l'absurde que $c_1 \neq c_2$.

Soient $p_1, \ldots, p_n$ les perles de c_1 et $q_1, \ldots, q_m$ les perles de c_2. Alors :

$$\begin{cases} t_1 = f(p_1) \cdot f(p_2) \cdot \ldots \cdot f(p_n) \\ t_2 = f(q_1) \cdot f(q_2) \cdot \ldots \cdot f(q_m) \end{cases} \qquad \text{où} \qquad \begin{cases} f(0) = a \\ f(1) = ab \\ f(2) = abb \end{cases}$$

Comme $c_1 \neq c_2$, on a trois cas :

→ c_1 est un préfixe de c_2. Alors $n \leqslant m$ et même $n < m$ car $c_1 \neq c_2$. Ainsi, t_1 est un préfixe de t_2 et comme $f(q_{n+1})$ existe et n'est pas vide : $t_1 \neq t_2$. C'est une contradiction.

→ c_2 est un préfixe de c_1. Par symétrie, on obtient une contradiction.

→ Sinon, il existe $i \in [\![1, \min(n, m)]\!]$ tel que $p_i \neq q_i$. Soit i_0 le plus petit indice vérifiant $p_{i_0} \neq q_{i_0}$. Pour tout $i \in [\![1; i_0 - 1]\!]$:

$$p_i = q_i \qquad \text{donc} \qquad f(p_i) = f(q_i)$$

Ainsi :

$$f(p_{i_0}) \cdot f(p_{i_0+1}) \cdot \ldots \cdot f(p_n) = f(q_{i_0}) \cdot f(q_{i_0+1}) \cdot \ldots \cdot f(q_m)$$

Comme p_{i_0} et q_{i_0} sont deux perles différentes : $(p_{i_0}, q_{i_0}) \in \{(0,1), (0,2), (1,0), (1,2), (2,0), (2,1)\}$. Traitons le cas $(p_{i_0}, q_{i_0}) = (0, 1)$ (les 5 autres cas sont similaires). On a :

$$a \cdot f(p_{i_0+1}) \cdot \ldots \cdot f(p_n) = ab \cdot f(q_{i_0+1}) \cdot \ldots \cdot f(q_m)$$

Donc :

$$f(p_{i_0+1}) \cdot \ldots \cdot f(p_n) = b \cdot f(q_{i_0+1}) \cdot \ldots \cdot f(q_m)$$

On en déduit que $f(p_{i_0+1})$ commence par un b ce qui constitue une contradiction (voir la définition de f).

Question 10.e –

```
(* Renvoie le plus petit indice j >= i tel que s.[j] = 'a'
   Si j n'existe pas, la fonction renvoie "String.length s"
   Hypothèse: i < String.length s *)
let prochain_a (s: mot) (i: int) =
  let j = ref i in
  while !j < String.length s && s.[!j] <> 'a' do
    incr j
  done;
  !j;;
```

```
let collierTM (n: int): collier =
  let s = motTM n in
  let rec aux (i: int) =
    if i = String.length s then [] else
      match prochain_a s (i+1) with
      | j when j = i+1 -> 0 :: aux j
      | j when j = i+2 -> 1 :: aux j
      | j when j = i+3 -> 2 :: aux j
      | _ -> failwith "collierTM: mot de Thue-Morse invalide"
  in
  aux 0;;
```

Question 11.a – Montrons par récurrence sur $n \in \mathbb{N}$ que pour tout $i \in [\![0, 2^n - 1]\!]$:

$$\begin{cases} s_{n,i} = a & \text{si } \alpha(i) \text{ est pair} \\ s_{n,i} = b & \text{si } \alpha(i) \text{ est impair} \end{cases}$$

→ Initialisation pour $n = 0$. Si $i \in [\![0; 2^0 - 1]\!]$, alors $i = 0$ et donc $s_{0,0} = a$. De plus, $\alpha(0) = 0$, d'où le résultat.

→ Hérédité. Soit $n \in \mathbb{N}$. Supposons la propriété vraie au rang n et montrons la au rang $n+1$. On remarque que tout $i \in [\![0; 2^{n+1} - 1]\!]$ s'écrit $i = 2j$ ou $i = 2j + 1$ avec $j = \lfloor i/2 \rfloor \in [\![0; 2^n - 1]\!]$. De plus, pour tout $j \in [\![0; 2^n - 1]\!]$, $s_{n+1,2j}$ est la première lettre de $\phi(s_{n,j})$ et $s_{n+1,2j+1}$ est la deuxième lettre de $\phi(s_{n,j})$. Ainsi :

- Si $s_{n,j} = a$, alors $\phi(s_{n,j}) = ab$, donc $s_{n+1,2j} = a$ et $s_{n+1,2j+1} = b$.
- Si $s_{n,j} = b$, alors $\phi(s_{n,j}) = ba$, donc $s_{n+1,2j} = b$ et $s_{n+1,2j+1} = a$.

De plus, pour tout $j \in [\![0; 2^n - 1]\!]$:

- Comme l'écriture en base 2 de $2j$ est l'écriture en base de j à laquelle on a concaténé un 0 à droite, on obtient $\alpha(2j) = \alpha(j)$.
- Comme l'écriture en base 2 de $2j + 1$ est l'écriture en base de j à laquelle on a concaténé un 1 à droite, on obtient $\alpha(2j) = \alpha(j) + 1$.

Nous sommes maintenant prêts pour montrer la propriété au rang $n + 1$. Pour tout $i \in [\![0, 2^{n+1} - 1]\!]$, on pose $j = \lfloor i/2 \rfloor$ et on traite quatre cas :

- i est pair et $\alpha(i)$ est pair. Alors $\alpha(j) = \alpha(i)$ est pair. Par l'hypothèse de récurrence, $s_{n,j} = a$ et donc $s_{n+1,i} = a$
- i est pair et $\alpha(i)$ est impair. Alors $\alpha(j) = \alpha(i)$ est impair. Par l'hypothèse de récurrence, $s_{n,j} = b$ et donc $s_{n+1,i} = b$
- i est impair et $\alpha(i)$ est pair. Alors $\alpha(j) = \alpha(i) - 1$ est impair. Par l'hypothèse de récurrence, $s_{n,j} = b$ et donc $s_{n+1,i} = a$
- i est impair et $\alpha(i)$ est impair. Alors $\alpha(j) = \alpha(i) - 1$ est pair. Par l'hypothèse de récurrence, $s_{n,j} = a$ et donc $s_{n+1,i} = b$

Question 11.b –

```
(* Renvoie la somme des bits de l'écriture en base 2 de i.
   Hypothèse: i >= 0. *)
let rec somme_bits (i: int) =
  if i = 0 then 0 else (i mod 2) + somme_bits (i/2);;
```

```
(* Renvoie 2**n.
   Hypothèse: n >= 0. *)
let rec puiss2 (n: int): int = match n with
  | 0 -> 1
  | n when n mod 2 = 0 -> let x = puiss2 (n/2) in x*x
  | n                  -> let x = puiss2 (n/2) in 2*x*x;;
```

```
let motTM_bis (n: int): mot =
  String.init
    (puiss2 n)
    (fun i -> if (somme_bits i) mod 2 = 0 then 'a' else 'b');;
```

Question 12.a – Pour tout mot u, on note $\psi(u)$ le mot obtenu en remplaçant chaque a par un b et inversement. Par exemple, $\psi(abbabaa) = baababb$.

On remarque que $\psi(\phi(a)) = \psi(ab) = ba = \phi(b) = \phi(\psi(a))$ et de même $\psi(\phi(b)) = \phi(\psi(b))$. On en déduit par une récurrence immédiate sur la taille d'un mot u que $\psi(\phi(u)) = \phi(\psi(u))$.
À l'aide de ce résultat, montrons par récurrence sur $n \in \mathbb{N}$ que $s_{n+1} = s_n \cdot \psi(s_n)$:

→ Initialisation pour $n = 0$. On a $s_1 = ab = s_0 \cdot \psi(s_0)$.

→ Hérédité. Soit $n \in \mathbb{N}$ tel que $s_{n+1} = s_n \cdot \psi(s_n)$. Alors :

$$\begin{aligned} s_{n+2} &= \phi(s_{n+1}) \\ &= \phi(s_n \cdot \psi(s_n)) && \text{par hypothèse de récurrence.} \\ &= \phi(s_n) \cdot \phi(\psi(s_n)) \\ &= \phi(s_n) \cdot \psi(\phi(s_n)) \\ &= s_{n+1} \cdot \psi(s_{n+1}) \end{aligned}$$

D'où le résultat.

Montrons maintenant par récurrence sur $n \in \mathbb{N}$ que $u_n = s_n$ et $v_n = \psi(s_n)$:

→ Initialisation pour $n = 0$. On a $u_0 = a = s_0$ et $v_0 = b = \psi(s_0)$.

→ Hérédité. Soit $n \in \mathbb{N}$ tel que $u_n = s_n$ et $v_n = \psi(s_n)$. Alors :

$$u_{n+1} = u_n \cdot v_n = s_n \cdot \psi(s_n) = s_{n+1} \quad \text{et} \quad v_{n+1} = v_n \cdot u_n = \psi(s_n) \cdot s_n = \psi(s_n \cdot \psi(s_n)) = \psi(s_{n+1})$$

Remarque. On aurait pu traiter cette question à l'aide du résultat de la question 11.a

Question 12.b –

```
(* Fonction qui renvoie le couple (u_n, v_n).
   Hypothèse: n >= 0 *)
let rec get_uv (n: int) = match n with
  | 0 -> "a", "b"
  | n -> let u,v = get_uv (n-1) in u^v, v^u;;
```

```
let motTM_ter (n: int) =
  let u,_ = get_uv n in
  u;;
```