

## Deug MIAS: UE SM23 Examen de TP Informatique Correction

### 1 Lecture d'expressions Caml (5 pts):

Pour chacune des expressions suivantes, donner la réponse de l'interpréteur Caml (type et valeur) quand elles sont correctes.

1. `float_of_int(3);;`

- : float = 3.0

2. `if 3 < 5 then 7.5  
    else 3.0 + 7.2;;`

Erreur il faut mettre +. car le + n'est défini que pour les entiers.

3. `let x = (3=4) in  
    if x then [(3,4)]  
    else [2];;`

Erreur, les deux branches d'un if-then-else doivent avoir le même type.  
Ici la première liste est de type `int*int` et la seconde de type `int`.

4. `(2,6)::[(4,3);(6,3);(7,5)]@[9,0];;`  
- : (int \* int) list = [2, 6; 4, 3; 6, 3; 7, 5; 9, 0]

5. `premier([fin([2;6;7;8]):[[7]]]);;`  
- : int list list = [[6; 7; 8]; [7]]

6. `let x=3 and y=x+2 in  
    (y*x,int_of_string("234"));;`

Erreur l'identificateur x n'est pas défini dans `y=x+2`.

7. `let x=((let z=3 and y=2 in y-z), 13.8) in  
    x::[];;`  
- : (int \* float) list = [-1, 13.8]

8. `let z=3 and x=4 and y=0 in  
    y = (let x=z+2 and y=x in x-y);;`

## 2 Gestion d'un club de sport (8 pts):

### 2.1 Questions

- Donner la définition du type Caml, SeqAdh représentant la séquence des informations des adhérents du club.

```
type SeqActiv == Nom_Activite list;;
type SeqAdh == (Nom_Personne * Date * SeqActiv) list;;
```

- Donner l'expression Caml représentant la séquence seq\_exemple.

```
[ "Nathanael", (12,10,2002), ["tennis";"piscine";"yoga"];
  "Ariane", (3,3,2003), ["danse";"Karate"];
  "Delphine", (16,6,2002), ["step"]; "Josua", (17,1,2003), ["muscultation"] ]
```

- Donner une fonction Caml, NbAdh, permettant de calculer le nombre d'adhérents du club.

```
let rec (NbAdh: SeqAdh -> int) = function
  [] -> 0
  | e::s -> 1 + NbAdh(s);;
```

- Donner une fonction Caml, Seq\_Kar, en terme d'une fonction intermédiaire que l'on donnera également, qui calcule la séquence des adhérents pratiquant le Karaté.

```
let rec (Est_Dans: SeqActiv * Nom_Activite -> bool) = function
  ([],act) -> false
  | (a::l,act) -> act = a || Est_Dans(l,act);;
```

```
let rec (Seq_Kar: SeqAdh -> Nom_Personne list) = function
  [] -> []
  | (n,a,l)::S -> if Est_Dans(l,"Karate") then n::Seq_Kar(S)
                  else Seq_Kar(S);;
```

- Donner une fonction Caml, Adh\_OK, qui prend une date, une séquence d'adhérents et calcule la séquence des adhérents qui auront le droit d'accéder au club après la date donnée en argument.

```
let (inf_date : Date*Date -> bool) =function
  (j1,m1,a1),(j2,m2,a2) -> if a1<a2 then true
                           else if a1>a2 then false
                           else if m1<m2 then true
                           else if m1>m2 then false
                           else j1<j2;;
```

```
let rec (Adh_OK: Date * SeqAdh -> SeqAdh) = function
  (d,[]) -> []
  | (d,(n,dt,activite)::S) -> if inf_date(dt,d) then Adh_OK(d,S)
                              else (n,dt,activite):: Adh_OK(d,S);;
```

### 3 Suite de Syracuse (7 pts):

La suite de Syracuse  $S_n(e)$  d'un entier positif  $e$  est définie par récurrence par les équations suivantes :

$$S_0(e) = e$$
$$S_n(e) = \begin{cases} S_{n-1}(e)/2 & \text{si } S_{n-1}(e) \text{ est pair.} \\ 3S_{n-1}(e) + 1 & \text{sinon} \end{cases} \quad \text{pour } n \geq 1$$

#### 3.1 Observation de la fonction Syr

Il y a une légère erreur dans la solution proposée. L'interpréteur répond.

Entree Interactive:

```
> | (n,e) -> if Syr(n-1,e) mod 2 = 0
>           ^^^
>
```

L'identificateur Syr n'est pas défini.

- Quelle est précisément cette erreur et comment la corriger ?

L'erreur provient de l'oubli du mot clef `rec` car la fonction est récursive. Pour la corriger on doit écrire :

```
let (Syr: int*int ->int)= fonction
  (0,e) -> e
  | (n,e) -> if Syr(n-1,e) mod 2 = 0
              then Syr(n-1,e)/2
              else 3*Syr(n-1,e)+1;;
```

- Indenter correctement la trace pour mettre en évidence la structure des appels à la fonction Syr.

```
#trace "Syr";;
#Syr(2,3);;
Syr <-- 2, 3
  Syr <-- 1, 3
    Syr <-- 0, 3
      Syr --> 3
    Syr <-- 0, 3
      Syr --> 3
  Syr --> 10
  Syr <-- 1, 3
    Syr <-- 0, 3
      Syr --> 3
    Syr <-- 0, 3
      Syr --> 3
  Syr --> 10
Syr --> 5
- : int = 5
```

- Combien d'appels récursifs sont engendrés par l'appel `Syr(2,3)`?

Il y a 6 appels récursifs à `Syr(2,3)`.

Généraliser : combien faut-il d'appels récursifs pour calculer le  $n^{\text{ième}}$  terme d'une suite de Syracuse en utilisant `Syr` ? Justifier.

**Cas de bases :** pour  $n = 0$ , il y a 0 appels récursifs, pour  $n = 1$  il y a deux appels récursifs et pour  $n = 2$  il y a  $2 + 2 * 2 = 6$  appels récursifs. **Conjecture :** pour  $n$  il y a  $2^{n+1} - 2$  appels récursifs. **Supposons que ce soit le cas pour le rang  $n$ . Pour le rang  $n + 1$  il faut  $2 + 2 * (2^{n+1} - 2) = 2 + 2^{n+2} - 4 = 2^{n+1} - 2$ . La propriété est montrée par induction.**

- La version proposée contient des calculs redondants. Expliquer pourquoi et comment cela se traduit sur la trace. Donner une deuxième version `Syr2` plus efficace.

```
let rec (Syr2: int*int ->int)= function
  (0,x) -> x
  | (n,x) -> let rep= Syr2 (n-1,x) in
              if rep mod 2 = 0
                then rep/2
                else 3*rep+1;;
```

## 3.2 Etude de la suite de Syracuse

- Donner une fonction Caml `Temps_Vol` qui calcule le temps de vol d'un entier.

```
let rec (Temps_Vol: int -> int) = function
  1 -> 0
  |2 -> 0
  |4 -> 0
  |x -> if x mod 2 = 0 then 1+Temps_Vol(n/2) else 1+ Temps_Vol(3*n+1);;
```

- Donner une fonction Caml `Alt_Max` qui calcule l'altitude maximale d'un entier.

```
let rec (Maxi: int*int->int) = function
  (mx,1) -> mx
  |(mx,2) -> mx
  |(mx,4) -> mx
  |(mx,n) -> let rep=(if n mod 2 = 0 then n/2 else 3*n+1) in
              if rep>mx then Maxi(rep,rep)
                else Maxi(mx,rep);;

let (Alt_Max: int->int)= function
  n -> Maxi(n,n);;
```