+1/1/60+

Questionnaire du QCM d'INF201, INF231, INF251

Répondre sur la feuille réponse à part

Quelques rappels

Quelques notations et fonction sur les listes

Etant donnée une liste non vide `l=[a1;a2;...;an]` les fonctions `List.hd` et `List.tl` donnent la tête et la queue de la liste:

```
(List.hd [a1;a2;...;an])=a1
(List.tl [a1;a2;...;an])=[a2;...;an]
```

Opérateurs d'ordre supérieur sur les listes

Nous rappelons ici le fonctionnement de quelques opérateurs d'ordre supérieur sur les listes.

- `(List.map f [a1;...;an])` vaut `[(f a1);...;(f an)]`.
- `(List.fold_left f init [a1;...;an])` vaut `(f ... (f (f init a1) a2) ... an)`.
- `(List.fold_right f [a1;...;an] init)` vaut `(f a1 (f a2 (...(f an init)...))`.
- `(List.filter p l)` est la liste des éléments de `l` qui satisfont `p`.
- `(List.exists p l)` vaut `true` ssi il existe un élément de `l` qui satisfait `p`.
- `(List.for_all p l)` vaut `true` ssi tous les éléments de `l` satisfont `p`.

Questions :

Question 1 ♣ Parmi les expressions suivantes, lesquelles causent une erreur de type ?

A `[1]::[[2];[3]]`	C `[1]@[2;3]`	■ `1::[[2];[3]]`	■ `1 @ [2;3]`
B `1::[2;3]`	D `[2;3]@[1]`	■ `[1]::[2;3]`	■ `[2;3]::1`

Question 2 ♣ On définit y par
`let x = true;; let y = (x=false) || (not x);;`
Donnez la valeur de y

A indéfinie	■ `false`
B `true`	

Question 3 ♣ Quelle est le type de la fonction `f` définie par `let f (x:int) (y:int): int = (x+y)/2;;`

■ `int→(int→int)`	C `int→int`	E `int`
■ `int→int→int`	D `int*int→int`	F `(int→int)→int`

Question 4 ♣ L'expression `if a then a else not a` est équivalente à

A ne peut pas se simplifier	B `not a`	C `false`	■ `true`	E `a`

Question 5 ♣ L'expression `if (a||b) then (a&&b) else b` est équivalente à

- ■ `a && b`
- B ne peut pas se simplifier
- C `(not a) || b`
- D `(not a) && b`
- E `not (a && b)`
- F `not (a || b)`
- G `a || b`

Question 6 ♣ Pour implémenter la fonction identité sur les listes je peux définir la fonction `id` par:

- A `let id (l:'a list):'a list=List.map (fun e -> [e]) l`
- ■ `let id (l:'a list):'a list=l`
- C Aucune des autres réponses ne sont correctes.
- D
```
let id (l:'a list):'a list=
List.fold_right (fun e s -> [e] ::  s) l []
```

Question 7 ♣ On définit la fonction `turn` par `let turn (x,y,z) = (z,x,y);;` Le type de `turn` est:

- A 'a * 'b *'c
- B non défini
- ■ 'a * 'b * 'c → 'c * 'a* 'b
- D 'a * 'a*'a
- E ('a → 'b → 'c) → ('c→'a → 'b)
- F 'a * 'a * 'a → 'a * 'a * 'a
- G unit

Question 8 ♣ Etant donnée une fonction `f` de type `int → int`, pour définir `g`, qui à x associe $(f(x))^2$ on peut écrire:

- ■
```
let square h =fun x->(h x)*(h x);;
let g = square f;;
```
- B `let g x= (f f) x`
- C `let g = f*f`
- D `let g x= f (f x)`
- ■ `let g x = (f x)*(f x);;`
- ■
```
let g = let square h x=(h x)*(h x)
in square f;;
```

Question 9 ♣ On définit
`let rec u n = if n=0 then 1 else 2*(u (n-1))+2;;`.
Que vaut `(u 3)` ?

A 16 ■ 22 E 24 G 10 I 20
B 14 D 30 F 26 H 18 J 12

Question 10 ♣ On définit la fonction `f` par
`let rec f (n:int) :  int = if n < = 1 then n else (f (n-1))+(f (n-2));;`

- A `(f 3)` vaut 3
- ■ `(f n)` est défini pour tout entier n
- C `(f (-1))` n'est pas défini.
- D `(f 3)` vaut 5
- ■ `(f 0)` vaut 0
- ■ `(f 2)` vaut 1

Question 11 ♣ On définit la fonction `f` par

```
let f (l:'a list):'a list = match l with
|[]-> []
|x::suite-> [x;x];;
```

- [ ] A `f` termine uniquement sur la liste vide
- [x] B `(f [3])` vaut `[3;3]`
- [ ] C `(f [3])` renvoie une erreur de typage.
- [ ] D `f` n'est pas définie car il manque `rec`
- [ ] E `(f [3])` ne termine pas.
- [ ] F Le filtrage (pattern matching) n'est pas exhaustif.

Question 12 ♣ Que vaut

```
List.fold_right (fun x b -> (x*List.hd(b))::b) [1;1;1] [1];;
```

- [ ] A `[4;3;2;1]`
- [ ] B `[1;4;9;16]`
- [ ] C `[1;2;4;8]`
- [ ] D `[1;2;3;4]`
- [x] E `[1;1;1;1]`
- [ ] F `[16;9;4;1]`
- [ ] G `[8;4;2;1]`

Question 13 ♣ Que vaut

```
List.fold_left (fun accu e ->if accu<5 then e else accu+e) 0 [2;4;6;5;3];;
```

- [ ] A 20
- [ ] B 6
- [ ] C 3
- [ ] D 4
- [ ] E 5
- [ ] F 12
- [x] G 14
- [ ] H 2

Question 14 ♣ On définit `f` par

```
let f x y=(y x);;
```

- [x] A y est forcément une fonction.
- [ ] B Le type de f est `'a->'b->'a`
- [ ] C Cette fonction ne peut pas être typée correctement
- [ ] D (f 2 3) est bien défini
- [x] E Le type de f est `'a->('a->'b)->'b`
- [x] F (f 3 (fun t->t+2)) est bien défini
- [x] G Le type de (f x y) n'est pas forcément le même que celui de x.

Question 15 ♣ Parmi les fonctions suivantes lesquelles peuvent être implémentées sans récursivité (ni appel à des fonctions récursives).

- [x] A `List.hd` (accès à la tête d'une liste)
- [x] B `List.tl` (accès à la queue d'une liste)
- [ ] C `List.filter`
- [ ] D `List.map`
- [ ] E `List.for_all`
- [ ] F la fonction qui renvoit le dernier élément d'une liste.
- [x] G la fonction qui renvoie le quatrième élément d'une liste pour les listes de longueur supérieure à 4 et 0 sinon.
- [ ] H `List.exists`
- [ ] I `List.fold_left`

Question 16 ♣ On définit la fonction `op` par

```
let rec op f l = match l with
   |[]-> []
   |x::suite-> (f x)@(op f suite);;
```

Quel opérateur sur les listes `op` implémente-t'il ?

- [A] List.exists
- [B] List.map
- [C] List.fold_right
- [D] List.fold_left
- [E] List.filter
- [F] List.for_all
- [■] Aucun opérateur dans cette liste

Question 17 ♣ On définit la fonction `f` par

```
let rec f (l:int list):int list = match l with
   |[]-> []
   |[x]-> [x]
   |x ::  y ::  suite-> y ::  (f (x ::  suite))
```

- [■] `f [2;1]` vaut `[1;2]`
- [B] `(f [3;2;1])` renvoie une erreur
- [■] `f` termine pour toute entrée
- [D] Le filtrage (pattern matching) n'est pas exhaustif.
- [E] `f` ne termine pas pour certaines entrées
- [F] `f` ne termine sur aucune entrée
- [■] `(f [])` vaut `[]`

+1/5/56+

Feuille Réponse du QCM d'INF201, INF231
A rendre avec votre copie

0 0 0 0 0
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
6 6 6 6 6
7 7 7 7 7
8 8 8 8 8
9 9 9 9 9

⟵ Codez votre numéro d'**anonymat** ci-contre, et écrivez le en dessous. N'écrivez **rien** d'autre.

Numéro d'anonymat:

.....

Veillez à bien noircir les cases au stylo (noir de préférence). Toutes les questions valent 3 points chacune et ont une bonne réponse ou plus. Le score obtenu pour chaque question ne peut pas être négatif. Une case noircie alors qu'elle ne devrait pas annule les points de la questions. Si telle n'est pas le cas le score de la question est le nombre de cases bien noircie sur le nombre de cases qu'il faut noircir.

Veillez à bien lire toutes les questions. Elles ne sont pas classées par ordre de difficulté croissante.

Q1 : A B C D
Q2 : A B
Q3 : C D E F
Q4 : A B C E
Q5 : B C D E F G
Q6 : A C D
Q7 : A B D E F G
Q8 : B C D
Q9 : A B D E F G H I J
Q10 : A C D
Q11 : A C D E F
Q12 : A B C D F G
Q13 : A B C D E F H
Q14 : B C D
Q15 : C D E F H I
Q16 : A B C D E F
Q17 : B D E F