+1/1/60+

Licence 1 **Devoir surveillé de INF201 et INF231, Partie QCM (à rendre à part)**

Sur 20 points. Durée conseillée 40 minutes. Cocher les cases correspondant à votre numéro d'étudiant à 8 chiffres (chaque chiffre est codé par une case cochée dans la colonne correspondante).

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

⟵ codez votre numéro d'étudiant ci-contre, et écrivez votre nom et prénom ci-dessous.

Nom et prénom :

.....

Les questions sont indépendentes et ne sont pas classées par ordre de difficultés. Toutes les questions ont une bonne réponse ou plus. Pour avoir les points d'une question il faut exactement cocher les réponses correctes. Aucune question n'entraîne de points négatifs.

Question 1 ♣ L'expression `if a then true else false` est équivalente à

- ☐ `a`
- ☐ ne peut pas se simplifier
- ☐ `not a`

Question 2 ♣ Le type `point` défini par `type point = float*float` est

- ☐ un type énuméré
- ☐ un sale type
- ☐ un type récursif
- ☐ un type produit
- ☐ un type somme
- ☐ un type synonyme du type `float*float`

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

- ☐ `float`
- ☐ `int`
- ☐ `int→int`
- ☐ `int→int→int`

Question 4 ♣ La fonction `f` définie par `let rec f (n: int):int = n*f (n-1)`

- ☐ renvoie la valeur n! (factorielle n)
- ☐ ne termine jamais, quelque soit l'entrée `n`
- ☐ est récursive

Question 5 ♣ Le type `t` défini par `type t = Empty | Node of t*t` est

- ☐ un type énuméré
- ☐ un type somme
- ☐ un brave type
- ☐ un type récursif
- ☐ un type produit

Question 6 ♣ On veut définir une fonction `sum : int → int` telle que `sum n` est la somme des entiers de `0` à `n`. Parmi les implémentations suivantes lesquelles sont correctes:

- ☐ `let sum n = (n*(n+1))/2`
- ☐
```
let rec sum n = if n>0
then n+(sum (n-1)) else 0
```
- ☐
```
let rec sum n = match n with
|n->n+(sum (n-1))
|0-> 0
```
- ☐ aucune des autres réponses ne sont correctes.
- ☐
```
let rec sum n = match n with
|0-> 0
|n->n+(sum (n-1))
```

Question 7 ♣ La fonction `f` définie par `let f a x = if a then x = 2 else x = 3` est de type

- ☐ `bool→int→ bool`
- ☐ `int`
- ☐ `bool→bool→ bool`
- ☐ `bool→int`
- ☐ `bool→int→ int`
- ☐ `bool→bool→ bool`
- ☐ `bool*int→bool`
- ☐ `bool`
- ☐ `bool*int→int`
- ☐ ne peut pas être typé correctement

Question 8 ♣ La valeur de l'expression `let x = 10 in let x = 3 in let y = x+4 in x+y` est

- ☐ 24
- ☐ 10
- ☐ 17

Question 9 ♣ On veut implémenter une fonction `f` qui convertit les `float` sans décimale après le point (comme `2.`) en `int` (par ex. `(f 2.) = 2`) et qui renvoie `-1` sinon (par ex. `(f 2.1) = -1`).

- ☐ aucune des autres réponses n'est correcte.
- ☐ `let f (x:float):int = if x = int_of_float x then x else -1`
- ☐ `let f (x:float):int = let y = int_of_float x in if x = y then y else -1`
- ☐
```
let f (x:float):int = let y = (int_of_float x) in
if x = (float_of_int y) then y else -1
```

Question 10 ♣ On définit la fonction `f` par `let f b z = if not b then 5. else z`

- ☐ `z` est de type `unit`
- ☐ `b` est de type `bool`
- ☐ `(f true 4.)` vaut 5.
- ☐ `z` est de type float
- ☐ `(f true 4.)` vaut 4.

Question 11 ♣ Quelle est le type de l'expression `(float_of_int 2)`

- ☐ `int`
- ☐ `float`
- ☐ `unit`
- ☐ `bool`

Question 12 ♣ On définit la fonction `chevauche` par
```
let chevauche (bi1,bs1 :int*int) (bi2,bs2 :int*int) :  bool =
(bi1<bi2 && bi2<bs1 && bs1<bs2) || (bi2<bi1 && bi1<bs2 && bs2<bs1)
```
Parmi les expressions suivantes lesquelles valent `true`.

- ☐ `chevauche (2,3) (4,5)`
- ☐ `chevauche (4,8) (3,5)`
- ☐ `chevauche (4,5) (5,8)`
- ☐ `chevauche (2,8) (4,5)`
- ☐ `chevauche (0,4) (3,5)`
- ☐ `chevauche (4,5) (2,3)`

Question 13 ♣ La valeur de l'expression `let x = 10 in let x = 3 and y = x+4 in x+y` est

- ☐ 24
- ☐ 10
- ☐ 17

Question 14 ♣ L'expression `if a then b else false` est équivalente à

- ☐ ne peut pas se simplifier
- ☐ `a || b`
- ☐ `a && b`
- ☐ `not (a && b)`
- ☐ `(not a) || b`
- ☐ `not (a || b)`
- ☐ `(not a) && b`

Question 15 ♣ On définit le type `intlist` par `type intlist = Nil | Cons of int*intlist`.
On définit la fonction `f` par
```
let rec f (l:intlist):intlist = match l with
|Nil-> Nil
|Cons(0,ll)-> Cons(0,f ll)
|Cons(x,ll)-> f (Cons(x-1,Cons(0,ll)))
```

- ☐ `(f Cons(3,Cons(2,Nil)))` vaut `Cons(0,Cons(0,Cons(0,Cons(0,Cons(0,Nil)))))`
- ☐ `f` termine uniquement sur la liste vide
- ☐ Le filtrage (pattern matching) n'est pas exhaustif.
- ☐ `f` termine uniquement sur la liste vide et sur les listes contenant que des 0.
- ☐ `(f l)` ne termine pas ssi la liste contient au moins un nombre négatif.
- ☐ `(f Cons(m,Cons(n,Nil))` est égale à `(f Cons(n,Cons(m,Nil))` pour tout $m, n \geq 0$.

Question 16 ♣ On définit
```
let rec u n = if n< = 1 then 1 else (u (n-1))+(v (n-1)) and v n = u (n-1)
```
A quoi est égale `(u 4)`:

- ☐ `(v 5)`
- ☐ 5
- ☐ 8
- ☐ 1
- ☐ 3
- ☐ `(u 3)+(v 3)`

Question 17 ♣ Soient `f` et `g` définies par `let f x = 2*x` and `let g y = y+6` . Quelle est la valeur de `g (g (f (g 9)))`

- ☐ 42
- ☐ 64
- ☐ 32
- ☐ 201
- ☐ 1024
- ☐ 2019

Question 18 ♣ L'expression `2<3<6`

- ☐ s'évalue a `false`
- ☐ s'évalue a `true`
- ☐ renvoie une erreur de type

Question 19 ♣ On veut modéliser les mois de l'année. On peut pour cela définir le type `mois` par

- ☐ `type mois = Jan*Fev*Mar*Avr*Jun*Jui*Aou*Sep*Oct*Nov`
- ☐ `type mois = Jan|Fev|Mar|Avr|Jun|Jui|Aou|Sep|Oct|Nov|Dec`
- ☐ `type mois = 1|2|3|4|5|6|7|8|9|10|11|12`
- ☐ `type mois (x:int) if 1 <= x <= 12`
- ☐ `type mois = int (*restreint au entier de 1 à 12*)`

Question 20 ♣ On définit le type `intlist` par `type intlist = Nil | Cons of int*intlist`. On définit la fonction `f` par

```
let rec f (l:intlist):intlist = match l with
|Nil-> Nil
|Cons(0,ll)-> Cons(0,Cons(0,f ll))
|Cons(1,ll)-> f ll
|_->l
```

- ☐ `(f Cons(1,Cons(1,Nil)))` vaut `C1,Nil)`
- ☐ `f` termine pour toute entrée.
- ☐ `(f Cons(2,Nil))` renvoie une erreur.
- ☐ `f` ne termine pas pour certaines entrées.
- ☐ `(f Cons(1,Cons(0,Cons(2,Nil))))` vaut `Cons(0,Cons(0,Cons(2,Nil)))`
- ☐ `(f Cons(2,Nil))` vaut `Cons(2,Nil)`