

I. Syntaxe de *Caml* pour les définitions et les appels de fonctions (6 points)

Dites si les expressions *Caml* suivantes sont syntaxiquement correctes et donner la réponse de *Caml* (c'est-à-dire la valeur pour les définitions ou appels de fonctions, le profil pour les appels de fonctions, les messages d'erreur, etc...).

1. (2 points) Définitions locales

```
let a = 2 in a * a;;
let b = a * a;;
let a = 2 in let b = a * a;;
let b = let a = 2 in a * a;;
```

2. (2 points) Fonctions à plusieurs variables

```
(function a -> function b -> function c -> a + b * c) 1 2 3;;
(function a -> function b -> function c -> a or b & c) 1 2 3;;
(function a -> function b -> function c -> a or b & c) false true true;;
(function a -> function b -> function c -> a or b & c) false true a=b;;
```

3. (2 points) Polymorphisme

```
let f x y z = x = 2 or y = z ;;
let f x y z = if true then if false then 1 else 2 else 3;;
let f x y z = if true then if false then x else y else z;;
let f x y z = "bonjour" ;;
```

II. Fonctions récursives (5 points)

```
let rec u = function 0 -> 0 | n -> n - u (n-1);;
```

1. (2 points) Si la fonction récursive précédente termine, donner la valeur résultant des appels de fonction suivantes en détaillant toutes les étapes des appels récursifs :

```
u 5;; u 7;; u 6;; u 8;;
```

Des appels récursifs précédents vous pouvez constater que, pour  $n$  pair,

$$u_n = [n - (n - 1)] + [(n - 2) - (n - 3)] + \dots + [2 - 1] = 1 + 1 + \dots + 1 = \frac{n}{2}$$

et que, pour  $n$  impair,

$$u_n = [n - (n - 1)] + [(n - 2) - (n - 3)] + \dots + [3 - 2] + 1 = 1 + 1 + \dots + 1 = \frac{n}{2} + 1$$

2. (3 points) Donner une version analytique non récursive de cette fonction (de préférence sans structure conditionnelle) et vérifier sur les exemples que les valeurs obtenues sont les bonnes.