



☞ La participation est évaluée sur 2 points.

**Exercice I - la récursivité (4 points environ) :**

Etant donné deux entiers a, b, on veut calculer la somme : S (n) qu'on peut définir par récurrence :

$$S_0 = 1$$

$$S(n+1) = a^n + b * S(n-1) \quad \text{Pour } n \geq 1$$

$$U_n = U_{n-1} + C * U_{n-2}, \quad \text{pour } n \geq 2.$$

1. (1 points) Ecrire une fonction **puissance X\_de\_n** qui calcule la puissance  $X^n$ .

2. (3 points) Ecrire une fonction récursive directement inspiré de la définition par récurrence ci-dessus. (Calcul de S (n)).

**Exercice II - les types (7 points environ) :**

Pour programmer la gestion du Deug 1mias, on est amené à spécifier un type étudiant constitué de **nom** (string), **année** (int), **age** (int), et **sexe** (bool).

1. (1 points) Définie le type étudiant.
2. (2 points) Ecrire une fonction permettant de créer un type étudiant.
3. (1 points) Ecrire les 4 fonctions d'accès permettant d'extraire les différents éléments de ce type étudiant que vous nommerez respectivement nom, age, et sexe. (chaque fonction sur **0.25 points**).

Dans cette partie, l'objectif est l'indexation des étudiants :

4. (1 points) Ecrire une fonction (**test\_si** : étudiant : bool) qui test si le l'étudiant cherché dans l'indexation correspond au nom : « PIOT » et au sexe « F comme féminin », elle renvoie True, sinon elle devrait renvoyer False.
5. (1 points) Ecrire une fonction (**céelle** : étudiant -> bool) qui test si, l'étudiant cherché correspond non seulement au nom : « PIOT », au sexe « F » calculée à la question précédente (en utilisant la fonction de la question 4) et à l'age d'étudiant (e) : 19, elle renvoie True, sinon elle renvoie False.
6. (1 points) Si, j'utilise la fonction **trouvée** en question 5, est-ce que l'étudiante suivante, dont les références sont : (nom : PIOT, année : 2004, age : 19 ans, sexe : M comme masculin), sera trouvée (c'est-à-dire la fonction renvoie True) ou, il ne sera pas trouvée (la fonction renvoie False) ?

### Exercice III - les listes (7 points environ) :

1. **(2 points)** Définir une fonction **enleve** de type 'a list -> 'a list qui prends deux listes l1 et l2 et renvoie la liste obtenue en supprimant de l2 toutes les occurrences de tous les éléments de l1.

Par exemple : enleve [1 ; 2] [3 ; 4 ; 1 ; 5 ; 3 ; 2 ; 6 ; 5 ; 7 ; 1] renvoie la liste

[3 ; 4 ; 5 ; 3 ; 6 ; 5 ; 7]

2. **(2 points)** Définir une fonction **truc** qui prend une liste d'entiers et calcule la somme et le produit de ces entiers. Si la liste est vide, la somme et le produit sont nuls.
3. **(2 points)** Définir une fonction prédicat **tousegaux** de type 'a list -> bool qui teste si tous les éléments d'une liste sont égaux.

4. **(1 points)** Donner le type et s'il y a lieu la valeur, des expressions suivantes en langage Caml :

- `2 :: (4 :: (6 :: []));;`
- `0 :: [2; 4; 6 ];;`
- `hd ([2; 4; 6 ]);;` (**hd est la fonction tête d'une liste**)
- `[2; 4; 6] @ [1; 3; 5];;`

(chaque fonction est sur 0.25 points).

**NB : pour la notation utilisée, elle est susceptible d'être modifiée en fonction des résultats obtenus**