Nom :	***Prénom :***	***N°étu :***

Durée de l'épreuve : 1h. Aucun document n'est autorisé. Les téléphones doivent être rangés. Lisez-bien les énoncés ! Vous rédigerez vos réponses sur cette feuille.
Pour les questions 2, 3, 4, 5, 6 vous devez dans votre réponse donner la signature de la fonction (le type de son/ses paramètre(s) et de son résultat).

Question 1 (3 points)

1. Indiquez les types et valeurs des expressions suivantes :

	Type de l'expression	Valeur de l'expression
List.hd [4 ;2] + 1		
(8/3) : : (8 mod 3) : : [ ]		
List.hd(["un" ;"deux"]) < "trois"		

2. Quelle est la signature (types du/des paramètre(s) et résultat) des fonctions :

	Signature de la fonction
`let f = function (x,y) → if y then x+1 else 2*x`	
`let h = function (x,y) → x + 2*List.hd(y)`	
`let g = fun x y → x < List.hd(y)`	

Question 2 (3 points)
Définissez en OCaml la fonction memeParite : $\mathbb{N} \times \mathbb{N} \longrightarrow$ Bool qui vérifie si deux entiers naturels sont de même parité, c'est à dire si ils sont tous les deux pairs ou tous les deux impairs.
Exemples : `memeParite(4,6)` vaut `true` ; `memeParite(3,7)` vaut `true` ; `memeParite(4,3)` vaut `false`.

Question 3 (3 points)
Définissez la fonction queDesUn : $\mathbb{N} \longrightarrow$ Bool qui teste si l'écriture décimale d'un entier naturel n'est composée que du chiffre 1.
Exemples : `queDesUn(1111)` et `queDesUn(1)` valent `true` ; `queDesUn(110)` et `queDesUn(8)` valent `false`.

Question 4 (4 points)
Définissez en OCaml la fonction `prodDiv` qui étant donnés deux entiers naturels non nuls `n` et `i`, calcule le produit des diviseurs de `n` inférieurs ou égaux à `i`.
Exemples : `(prodDiv 45 7)` vaut 15 car les diviseurs de 45, inférieurs ou égaux à 7, sont 1, 3, et 5.
`(prodDiv 12 1)` vaut `1` ; `(prodDiv 12 2)` vaut `2` ; `(prodDiv 12 3)` vaut `6` ; `(prodDiv 12 5)` vaut `24`.

Question 5 (3 points)
Définissez la fonction `somElem` qui calcule la somme des éléments d'une liste d'entiers.
Exemples : `somElem [6 ; 9 ; 5]` vaut `20` ; `somElem [3]` vaut `3` ; `somElem [ ]` vaut `0`.

Question 6 (4 points)
Définissez en OCaml la fonction `prefLong` qui étant donnés une liste `li` et un entier naturel `n`, calcule le préfixe de longueur `n` de la liste `li`. Dans le cas où la longueur de `li` est inférieure à `n` le résultat est `li`.
Exemples :
`(prefLong [3;2;4] 0)` vaut `[ ]` ; `(prefLong [3;2;4] 1)` vaut `[3]` ; `(prefLong [3;2;4] 2)` vaut `[3;2]` ; `(prefLong [3;2;4] 3)` vaut `[3;2;4]` ; `(prefLong [3;2;4] 5)` vaut `[3;2;4]`.