Contrôle continu - IAP 1

mardi 18 octobre 2011 - 30 minutes

Nom - Prénom : .....

Complétez la session suivante. Il n'y pas d'erreur syntaxique mais des erreurs de typage peuvent exister. Dans ce cas, indiquez la réponse *erreur de typage* et expliquez pourquoi. Quand il y a deux phrases en `Ocaml`, on attend évidement deux réponses.

Répondez directement sur la feuille.

Echauffement : session OCaml

```
# let g x y = y && (x>0);;
```

```
# let h (x, y) = y && (x>0);;
```

```
# h(true,1);;
```

```
# let pi = 3.14 in (pi*.pi)/.6.;;
# pi;;
```

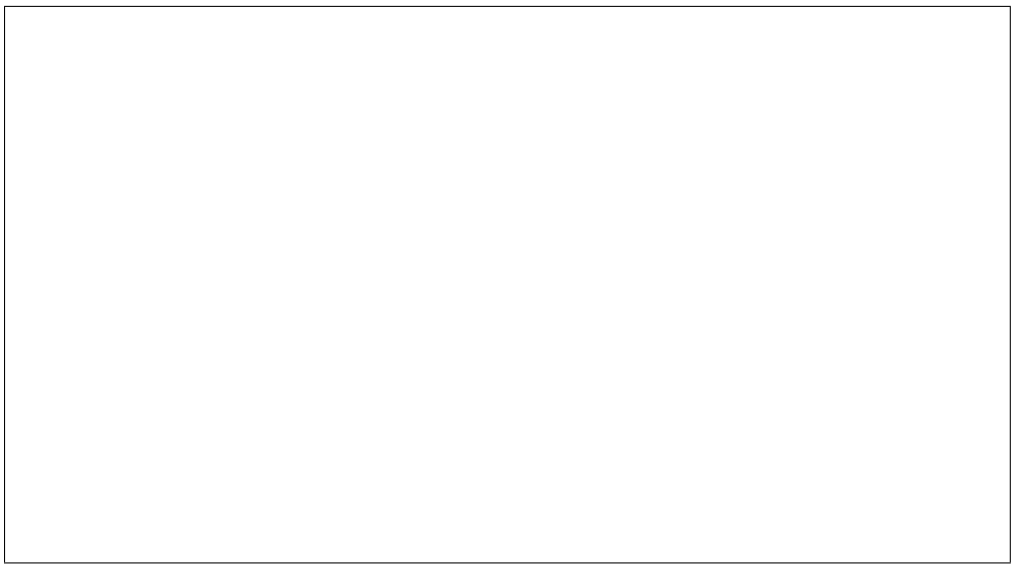
```
# let rec f l = match l with [] -> 0 | (a,b)::r -> if b then a+ (f r) else f r;;
```

Fonction 1 : Programmez la fonction dont l'interface est la suivante

```
(* interface second_degre
type : (float,float,float) -> float list
argument : a,b,c tel que ax^2 + bx + c = 0
precondition : **penser a bien traiter tous les cas particuliers**
postcondition : retourne la liste de toutes les solutions reelles de l'equation
tests : second_degre [1.0;-2.0;1.0] = [1.0] second_degre = [1.0;1.0;1.0] = [] *)
```

Fonction 2 : Programmez la fonction dont l'interface est la suivante

```
(* interface quotient_reste
type (int * int) -> (int * int)
arguments :  2 entiers a et b
précondition :b le diviseur est non nul
postcondition : produit un couple quotient/reste par soustractions successives
test(s) : quotient_reste (5,2) = (2,1) quotient_reste (10,3) = (3,1) *)
```



Fonction 3 : Programmez la fonction dont l'interface est la suivante

```
(* interface return_last_n
arguments : l,n
précondition : **quand la liste a moins de n elements, on renvoie tous les elements**
postcondition : retourne les n derniers éléments de la liste l
test(s) : return_last 2 [1.0;0.0; 2.0] = [0.0; 2.0]
*)
```

Fonction 4 : Ecrire l'interface de la fonction suivante

```
# let f4 x =
  let rec f4_aux x n = if (n*n)<=x then f4_aux x (n+1) else (n-1) in f4_aux x 0;;
```