

Nom	
Prénom	
Groupe	

Note	/ 4
------	-----

Algorithmique Contrôle 1 - Partie 1

INFO-SUP S1

EPITA

Oct. 2019

-
- ☐ Ceci est la partie 1 de l'épreuve - Vous devez rendre les deux parties !
 - ☐ Vous devez répondre directement **sur ce sujet**.
 - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées**.
 - Aucune réponse au crayon de papier ne sera corrigée.
 - ☐ La présentation est notée.
-

Exercice 1 (Un peu de cours... – 4 points)

1. Que retourne une opération interne ?

2. Comment appelle-t-on une opération servant à spécifier le domaine de définition d'un autre ?

3. Quels problèmes se posent lors de la conception de l'ensemble des axiomes ?

4. Quelles zones constituent la signature d'un type abstrait ?

5. Comment écrit-on des axiomes ?

Nom	
Prénom	
Groupe	

Note	/ 16
------	------

Algorithmique Contrôle 1 - Partie 2

INFO-SUP S1

EPITA

Oct. 2019

-
- ☐ **Ceci est la partie 2 de l'épreuve - Vous devez rendre les deux parties !**
 - ☐ Vous devez répondre directement **sur ce sujet**.
 - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées**.
 - Aucune réponse au crayon de papier ne sera corrigée.
 - ☐ **CAML :**
 - Tout code CAML non indenté ne sera pas corrigé.
 - En l'absence d'indication dans l'énoncé, les seules fonctions que vous pouvez utiliser sont **failwith** et **invalid_arg** (aucune autre fonction prédéfinie de CAML).
 - Tout code CAML doit être suivi du résultat son évaluation : la réponse de CAML.
 - ☐ La présentation est notée.
-

Exercice 2 (Dominos – 4 points)

Pour cet exercice, on s'intéresse au jeu de Dominos :

- Un domino sera représenté par un couple d'entiers (a, b)
- Une suite de dominos (chaîne) sera donc une liste de couples d'entiers $[(a_1, b_1); (a_2, b_2); \dots; (a_n, b_n)]$.

On considère qu'une chaîne de dominos est valide si les parties voisines des dominos ont le même nombre de points (voir les exemples ci-dessous).

Écrire la fonction `is_dominos` qui vérifie si une chaîne de dominos est valide.

Exemples d'applications :

```
# is_dominos [] ;;
- : bool = true
# is_dominos [(1,2); (2,3); (3,3); (3,6)] ;;
- : bool = true
# is_dominos [(2,3); (2,4); (1,4)] ;;
- : bool = false
```


Écrire la fonction `remove_nth` *i list* qui supprime la valeur à la $i^{\text{ème}}$ place dans la liste *list*. La fonction devra déclencher une exception `Invalid_argument` si *i* est négatif ou nul, ou une exception `Failure` si la liste est trop courte.

```
# remove_nth 5 [1; 2; 3; 4; 5; 6; 7; 8; 9] ;;
- : int list = [1; 2; 3; 4; 6; 7; 8; 9]

# remove_nth 10 [1; 2; 3; 4; 5; 6; 7; 8; 9] ;;
Exception: Failure "out of bound".

# remove_nth (-2) [1; 2; 3; 4; 5; 6; 7; 8; 9] ;;
Exception: Invalid_arg "negative rank".
```

[illegible]

1. Écrire la fonction CAML `for_all2` dont les spécifications sont les suivantes :
 - Elle prend en paramètre une fonction booléenne (un prédicat) à deux paramètres : `p` ainsi que deux listes : $[a_1; a_2; \dots; a_n]$ et $[b_1; b_2; \dots; b_n]$.
 - Elle vérifie si toutes les paires $a_i \ b_i$ vérifient le prédicat p .
 - Si elle trouve une paire telle que $p \ a_i \ b_i$ est faux, elle retourne faux. Sinon, elle déclenche une exception `Invalid_argument` si les deux listes sont de longueurs différentes.

[illegible]

- Exemples d'utilisation :*

```
# almost [1; 2; 3; 4; 5] [1; 4; 3; 4; 5];;
- : bool = false
```

[illegible]

Exercice 5 (Mystery – 2 points)

Donner les résultats des évaluations successives des phrases suivantes.

```
# let mystery a b =  
  let rec what = function  
    ([], _) -> true  
    | (_, []) -> false  
    | (e::l1, f::l2) -> (e = f) && what (l1, l2)  
  in  
  let rec is_that x y = match y with  
    [] -> 0  
    | e::q -> (if what (x, y) then 1 else 0) + (is_that x q)  
  in  
  is_that a b ;;
```

```
# mystery [1; 2] [1; 2] ;;
```

```
# mystery [1; 2] [1; 1; 2; 3; 3; 1; 2; 3] ;;
```

```
# mystery [1; 2] [2; 1] ;;
```

