Nom	
Prénom	
Groupe	
Prof TD	

Note	/15
------	-----

Algorithmique Contrôle 1 - Partie 2

Info-sup S1 Epita

4 Nov. 2021 - 10:00

Remarques (à lire!):

- □ 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 ou au stylo rouge 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 (Suppression intervalle – 6 points)

Écrire la fonction remove_interval list~i~j qui supprime toutes les valeurs comprises entre les places i et j dans la liste list (base 1). Toutes les valeurs aux places k avec $k \in [i,j]$ devront donc être supprimées de la liste list.

La fonction devra déclencher une exception :

- Invalid_argument si i ou j est négatif ou nul, ou si i > j
- Failure si la liste est trop courte.

 $Exemples\ d'applications\ avec\ {\tt list1}\ la\ liste\ [{\tt 11;\ 2;\ 23;\ 14;\ 5;\ 26;\ -7;\ 18;\ 99}]:$

```
# remove_interval list1 1 4;;
- : int list = [5; 26; -7; 18; 99]

# remove_interval list1 5 8;;
- : int list = [11; 2; 23; 14; 99]

# remove_interval list1 6 9;;
- : int list = [11; 2; 23; 14; 5]

# remove_interval list1 5 5;;
- : int list = [11; 2; 23; 14; 26; -7; 18; 99]

# remove_interval list1 9 6;;
Exception: Invalid_argument "remove_interval: invalid ranks".

# remove_interval list1 5 12;;
Exception: Failure "remove_internal: list too short".
```



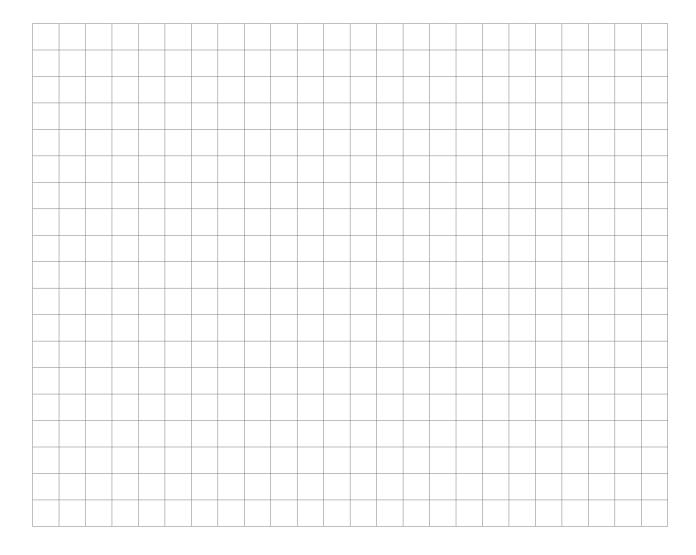
Exercice 3 (Listes alternées - 5 points)

Ecrire la fonction alternate_list l1 l2 avec la liste $l1 = [a_1; a_2; ...; a_n]$ et la liste $l2 = [b_1; b_2; ...; b_m]$ qui construit la liste alternée définie de la manière suivante :

- si les deux listes sont de même longueur, on alterne les éléments : voir exemple 1 (* ex 1 *);
- si la liste l1 est plus longue (n > m), on alterne jusqu'à finir l2 (il manquera donc des éléments de l1) : voir exemple 2;
- si la liste l2 est plus longue (m>n), l1 est "répétée" jusqu'à finir l2: $[a_1;b_1;a_2;b_2;...;\mathbf{a_n};\mathbf{b_n};a_1;b_{n+1};...;\mathbf{a_k};\mathbf{b_m}]$: voir exemples 3 et 4.

Exemples d'applications :

```
(* ex 1 *)
   # alternate_list ["a1"; "a2"; "a3"; "a4"; "a5"] ["b1"; "b2"; "b3"; "b4"; "b5"];;
   -: string list = ["a1"; "b1"; "a2"; "b2"; "a3"; "b3"; "a4"; "b4"; "a5"; "b5"]
(* ex 2 *)
   # alternate_list ["a1"; "a2"; "a3"; "a4"; "a5"] ["b1"; "b2"; "b3"];;
   - : string list = ["a1"; "b1"; "a2"; "b2"; "a3"; "b3"]
(* ex 3 *)
   # alternate_list ["a1"; "a2"; "a3"] ["b1"; "b2"; "b3"; "b4"; "b5"];;
   - : string list = ["a1"; "b1"; "a2"; "b2"; "a3"; "b3"; "a1"; "b4"; "a2"; "b5"]
(* ex 4 *)
   # alternate_list ["a1"; "a2"] ["b1"; "b2"; "b3"; "b4"; "b5"] ;;
   - : string list = ["a1"; "b1"; "a2"; "b2"; "a1"; "b3"; "a2"; "b4"; "a1"; "b5"]
  # alternate_list [] ["b1"; "b2"];;
  - : string list = ["b1"; "b2"]
  # alternate_list ["a1"] [];;
  - : string list = []
```



CONTRÔLE 1 - Part. 2- Nov. 2021 Exercice 4 (Mystery - 4 points)

Si elles sont incorrectes indiquez "Erreur".

Donner les résultats des évaluations successives des phrases suivantes si elles sont correctes.