Partie 0 : structure de files

Une structure de file est une structure mémorisant des éléments par leur ordre d'insertion dans la file dont on doit pouvoir retirer l'élément le plus ancien inséré dans la file. C'est une structure dite de type FIFO (first in first out). Une liste permet facilement d'insérer un nouvel élément mais pas de retirer l'élément le plus ancien.
On va implémenter une file par deux listes : une liste "début" et une liste "fin". Les éléments de "début" seront les premiers éléments de la liste, ceux de "fin" seront ceux de la fin, en commençant par le dernier.
On utilisera le type :
`type 'a file = { mutable debut : 'a list ; mutable fin : 'a list};;`

Par exemple `{ debut = [1;4] ; fin = [2; 3] }` est la file dont le premier élément est 1, le deuxième 4, le troisième 3 et le quatrième 2. Un nouvel élément devra être ajouté à la suite de 2. Le prochain élément à retirer sera 1.
Les files `{ debut = [1;4;3;2] ; fin = [] }` et `{ debut = [] ; fin = [2; 3; 4; 1]}` représenteront aussi cette file.

De façon générale, pour ajouter un élément à une file, on l'ajoutera en tête de la liste "fin". Pour retirer l'élément du début de la file, si "début" n'est pas vide, on retirera l'élément en tête de "début" ; si "début" est vide mais pas "fin", on modifiera "début" en le miroir de "fin" et "fin" en la liste vide et on retirera l'élément de tête du nouveau "début" de la file.

Question 1
Ecrire une fonction `creer_file_vide : unit -> 'a file` qui crée une file vide.

Question 2
Ecrire une fonction `est_file_vide : 'a file -> bool` indiquant si une file est vide.

Question 3
Ecrire une fonction `insere_file : 'a -> 'a file -> unit` qui modifie une liste en y ajoutant à la fin l'élément en argument.

Question 4
Ecrire une fonction `retire_file : 'a file -> 'a` qui renvoie le premier élément de la file s'il existe et une erreur sinon.

Question 5
Donner la complexité des fonctions précédentes. On donnera la complexité dans le pire cas et la complexité en moyenne, en considérant qu'une file de taille n a autant de probabilité d'être représentée par "début" de taille k et "fin" de taille $n-k$ pour tout $0 \leq k \leq n$.

Dans la suite du TP, on pourra utiliser le type `file` ou le module `Queue`.