Tramway

La CTS (Compagnie des Transports Strasbourgeois) vous charge de modéliser un tramway avec des passagers qui montent et qui descendent à chaque arrêt. Le code `tramway.ml` fourni contient deux listes de passagers qui serviront dans les exercices 3 et 5. Pensez à tester toutes vos fonctions avant de passer à la question suivante.

Exercice 1 : .

La ligne de tram.

1. Un arrêt de tram est représenté par une lettre (type `char`). Une ligne de tram est une liste d'arrêts, dont la tête est l'arrêt courant et la queue contient les arrêts à relier. Définissez une variable `ligne` qui contient les arrêts de 'A' à 'F'.
2. Écrivez une fonction `est_vide` qui teste si une liste est vide.
3. Écrivez une fonction `tete` qui renvoie la tête d'une liste, et lève une erreur si elle est vide.
4. Écrivez une fonction `queue` qui renvoie la queue d'une liste, et lève une erreur si elle est vide.
5. Écrivez une fonction `terminus` de prototype `char list -> bool` qui renvoie `true` si vous êtes au terminus (il n'y a qu'un arrêt dans la liste), `false` sinon.
6. Écrivez une fonction `avancer` de prototype `char list -> char list` qui enlève l'arrêt courant. Cette fonction doit lever une erreur s'il n'y a plus d'arrêt ou si vous êtes déjà au terminus.
7. Écrivez une fonction `incident` de prototype `char -> char list -> char list` qui prend en paramètres un arrêt `a` et une liste d'arrêts. Elle modélise un incident à l'arrêt `a` : elle renvoie la liste des arrêts tronquée après l'arrêt `a`. Il se peut que l'arrêt où se trouve l'incident ne soit pas dans la liste (déjà dépassé).

Exercice 2 : .

Les tarifs.

1. La CTS propose 5 tarifs : tarif 1 (plein tarif) à 1€40 ; tarif 2 (A-R) à 1€30 ; tarif 3 (carnet) à 1€20 ; tarif 4 (réduit) à 0€90 ; tarif 5 (abonnement) à 0€70. Définissez une variable `listetarifs` de type `float list` qui contient les 5 tarifs dans l'ordre.
2. Écrivez une fonction `prix` de prototype `int -> float list -> float` qui prend en paramètres un entier `i` et une liste de tarifs, et qui renvoie le `i`-ème tarif. Elle doit lever une erreur s'il n'existe pas de `i`-ème tarif.

Exercice 3 : .

Les usagers.

1. Un usager est défini par un quadruplet contenant une chaîne de caractères (son nom), deux caractères (son arrêt de départ et son arrêt d'arrivée), et un entier entre 1 et 5 (son tarif). Définissez le type `usager`, son constructeur `creer_usager`, et ses accesseurs `acces_nom`, `acces_d`, `acces_a` et `acces_t`.

2. Définissez une fonction `compter` de prototype `usager list -> int` qui compte le nombre d'usagers dans la liste. Testez-la avec la liste `usagers_aller` qui vous est fournie.
3. Définissez une fonction `compter_abonnes` de prototype `usager list -> int` qui compte le nombre d'abonnes dans une liste d'usagers.
4. Définissez une fonction `somme_des_prix` de prototype `usager list -> float` qui calcul la somme des prix payés par les usagers de la liste.

Exercice 4 :

Le tramway.

1. Un tramway est défini comme un triplet de listes : une liste de caractères (la liste des arrêts); une liste d'usagers (ceux qui sont dans le tram); une autre liste d'usagers (ceux qui attendent le tram). Définissez le type `tramway`, son constructeur `creer_tramway`, et ses accesseurs `acces_ligne`, `acces_dedans` et `acces_dehors`.
2. Définissez une variable `tram_vide` de type `tramway` qui est en début de ligne, avec aucun usager dedans, et tous les usagers de `usagers_aller` dehors.
3. Définissez une variable `tram_plein` de type `tramway` qui est en début de ligne, avec tous les usagers de `usagers_aller` dedans, et aucun usager dehors.
4. Écrivez une fonction `compter_passagers` de prototype `tramway -> int` qui compte les usagers qui sont dans le tram.
5. Écrivez une fonction `avancer_tram` de prototype `tramway -> tramway` qui renvoie le tramway passé en paramètre ayant avancé d'un arrêt.
6. Écrivez une fonction `monter` de prototype `tramway -> tramway` qui fait monter dans le tram tous les passagers qui attendent dehors à l'arrêt courant (c'est-à-dire dont l'arrêt de départ est égal à l'arrêt courant).
7. Écrivez une fonction `descendre` de prototype `tramway -> tramway` qui fait met dehors tous les passagers qui sont dedans et qui doivent descendre à l'arrêt courant.

Exercice 5 :

Les aller-retour. On veut maintenant que le tram puisse faire des aller-retour. Pour cela on définit le *sens* de parcours comme un booléen qui vaut `true` pour un parcours des arrêts dans l'ordre alphabétique, `false` dans l'ordre inverse.

1. Écrivez une fonction `sens_usager` de prototype `usager -> bool` qui détermine le sens du trajet d'un usager.
2. Écrivez une fonction `sens_tram` de prototype `tramway -> bool` qui détermine le sens du trajet d'un tramway. Cette fonction doit lever une erreur si le tramway est au terminus.
3. Écrivez une fonction `demitour` de prototype `char list -> char list` qui retourne une liste d'arrêts.
4. Écrivez une fonction `avancer_ar` de prototype `char list -> char list` similaire à la fonction `avancer` du premier exercice, à une nuance près : au terminus le tram fait demi-tour.
5. Construisez un tramway nommé `tram`, vide et en début de ligne, avec dehors tous les usagers de la liste `usagers_aller_retour` (fournie dans le fichier).

6. Écrivez une fonction `avancer_tram_ar` de prototype `tramway -> tramway` similaire à celle de l'exercice 4, mais qui prend en compte le demi-tour en fin de ligne.
7. La fonction `descendre` n'a pas besoin d'être modifiée, car un passager descend à son arrêt quel que soit le sens. En revanche il ne doit monter dans le tram que s'il va dans son sens. Écrivez une fonction `monter_ar` de prototype `tramway -> tramway` qui fait monter les passagers au bon arrêt dans le bon sens.