Matthieu Boyer
Planche A

USA USA USA

1. Définir un type `grade` qui est un type somme avec les variantes suivantes :
   — `Letter` contenant un caractère ;
   — `Dropped` contenant un booléen.
2. Écrire une fonction `grade_to_gpa : grade -> float option` qui convertit une note en moyenne générale (A=4.0, B=3.0, C=2.0, D=1.0, F=0.0). Renvoyer `None` pour les notes `Dropped`.
3. Écrire une fonction `is_passing : grade -> bool` qui détermine si une note est une réussite (C ou mieux pour les lettres, vrai pour PassFail). On considérera Incomplete et Withdrawn comme des échecs.
4. Écrire une fonction `grade_comparison : grade -> grade -> string` qui compare deux notes et retourne `"better"`, `"worse"` ou `"incomparable"`.
5. Écrire une fonction `average_grade : grade list -> float option` qui renvoie la lettre moyene d'un élève. On renverra `None` quand l'élève n'aura validé aucun cours. On se concentrera surtout sur le fait d'écrire une logique de fonction correcte

Montage Gla-Gla

1. Définir un type enregistrement `date` avec les champs suivants :
   - `day` de type entier
   - `month` de type entier
   - `year` de type entier
2. Écrire une fonction `is_leap_year : int -> bool` qui détermine si une année est bissextile (divisible par 4, sauf les multiples de 100 qui ne sont pas multiples de 400).
3. Écrire une fonction `days_in_month : int -> int -> int` qui prend un mois (1-12) et une année, et retourne le nombre de jours dans ce mois (28, 29, 30 ou 31).
4. Écrire une fonction `is_valid_date : date -> bool` qui vérifie si une date est valide (jour entre 1 et le nombre de jours du mois, mois entre 1 et 12).
5. Écrire une fonction `compare_dates : date -> date -> string` qui compare deux dates et retourne `"before"`, `"after"` ou `"equal"`.
6. Écrire une fonction `days_between : date -> date -> int option` qui calcule le nombre de jours entre deux dates. Retourner `None` si l'une des dates n'est pas valide. On se concentrera surtout sur le fait d'écrire une logique de fonction correcte.

Encore, et toujours, de la géométrie

1. Définir un type `point` qui est un tuple de deux flottants représentant les coordonnées (x, y) d'un point du plan euclidien.
2. Définir un type somme `shape` avec les variantes suivantes :
   — `Circle` contenant un enregistrement avec son centre et son rayon ;
   — `Rectangle` contenant un enregistrement avec son coin supérieur gauche, sa hauteur et sa largeur ;
   — `Triangle` contenant un enregistrement avec ses trois sommets.
3. Écrire une fonction `area : shape -> float` qui calcule l'aire de n'importe quelle forme. Pour le triangle, utiliser la formule de Héron ou la formule par coordonnées.
4. Écrire une fonction `perimeter : shape -> float` qui calcule le périmètre de n'importe quelle forme. Vous aurez besoin d'une fonction auxiliaire pour calculer la distance entre deux points.
5. Écrire une fonction `contains_origin : shape -> bool` qui détermine si la forme contient le point (0.0, 0.0).
6. Écrire une fonction `scale : shape -> float -> shape` qui met à l'échelle une forme par un facteur donné (multiplier toutes les dimensions par le facteur). Garder le même centre/position pour les cercles et les rectangles.