L1 MISPIC
Programmation fonctionnelle
TP 2

Avant de commencer le TP, créez un répertoire `TP2` dans votre arborescence consacrée à OCaml. Allez dans ce répertoire et ouvrez un fichier `tp2.ml` avec *emacs*. Vous taperez vos fonctions dans ce fichier, et les chargerez dans l'interpréteur OCaml avec `#use "tp2.ml";;` Si une fonction est erronée, corrigez-la dans *emacs*, puis rechargez le fichier dans l'interpréteur. Ceci vous permettra de pouvoir sauvegarder votre travail dans le fichier `tp2.ml`. Gardez à l'esprit que même si la définition d'une fonction ne fait pas d'erreur dans l'interpréteur, cela ne signifie pas pour autant qu'elle calcule ce que vous souhaitez. Vous devez pour vous en assurer la tester en l'appelant avec différentes valeurs de paramètres pour être sûr que ce qu'elle vous renvoie est correct.

1 Des notes

Un étudiant d'une autre université souhaite que vous l'aidiez à effectuer un ensemble de calculs sur les notes qu'il a obtenues au 1er semestre. Comme vous vous y connaissez en OCaml, vous décidez de programmer les calculs qu'il vous demande. Il commence par vous donner l'ensemble des matières qu'il a étudiées, ainsi que leurs coefficients :

Filière	Mathématiques	Physique	Chimie	Informatique	Total
Info	5	3	3	7	18
Math	7	3	3	5	18

1. Il vous apprend que les moyennes de Physique et de Chimie dépendent chacune de 2 notes, de même coefficient. Vous décidez donc d'écrire une fonction `moyenne_2_notes : float -> float -> float` retournant la moyenne de 2 notes. Cette fonction pourra être utilisée indifféremment par la suite pour calculer la moyenne de votre ami en Physique et en Chimie.
2. Pour les Mathématiques, votre ami a eu 3 notes de même coefficient. Écrire une fonction `moyenne_3_notes : float -> float -> float -> float` qui renvoie la moyenne correspondante.
3. En informatique, les choses sont un peu plus compliquées, puisqu'il a eu 3 notes, mais de coefficients différents. Sa première note, théorique, est de coefficient 3. Ses deux autres notes, pratiques, sont toutes les deux de coefficient 1. Écrire une fonction `moyenne_informatique : float -> float -> float -> float` qui renvoie la moyenne de cette matière.
4. Votre ami a donc au total reçu 10 notes : 2 en Physique, 2 en Chimie, 3 en Mathématiques, 3 en Informatique. Cependant, le calcul de sa moyenne

générale dépend de sa filière (voir le tableau ci-dessus). Comme il ne sait pas encore quelle filière choisir, vous lui proposez de calculer sa moyenne générale dans les 2 cas, pour comparer. Vous écrivez donc une fonction `moyenne_generale` qui prend en paramètres un caractère correspondant à la filière ('I' ou 'M') et 10 notes (dans l'ordre du tableau), et qui retourne la moyenne générale. Si la filière est inconnue, votre fonction générera une erreur.

5. Les notes de votre ami sont, dans l'ordre et sur 20 : 10, 6, 9, 11, 12, 9, 12.5, 8, 15, 14. Quelle filière lui conseillez-vous de prendre ?
6. Votre ami ne vous écoute pas, et, pensant qu'il peut faire mieux, décide de choisir la filière Math. Il vous demande alors quelle note il doit avoir au rattrapage de Mathématiques pour avoir 10 de moyenne générale. Le rattrapage donne une note, qui remplace les 3 qu'il avait eues au long du semestre. Vous écrivez donc une fonction `rattrapage_mathematiques` qui prend en paramètres ses 7 notes des autres matières et renvoie la note nécessaire au rattrapage de Mathématiques.
7. Se sentant capable de faire encore mieux, votre ami vous demande alors la note nécessaire au rattrapage de Mathématiques pour avoir 11 de moyenne générale. Puis pour avoir 12. Comprenant qu'il peut vous poser cette question pour n'importe quelle moyenne générale, vous décidez de modifier votre fonction précédente pour prendre en paramètre la valeur souhaitée en moyenne générale. Écrire la fonction `rattrapage_mathematiques_2` qui répond à ce problème.

2 Boîte de nuit

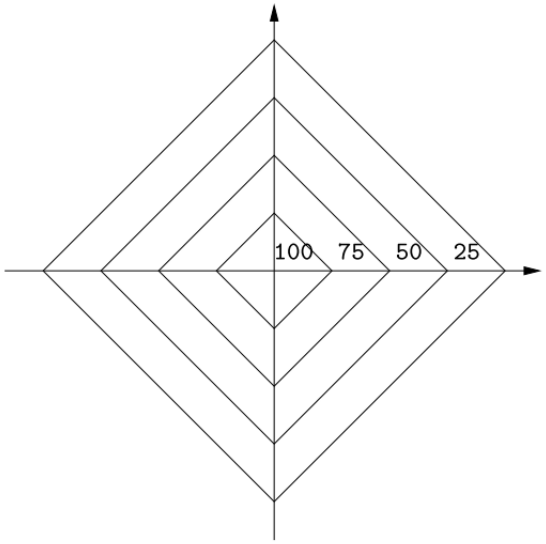
1. Le propriétaire d'une boîte de nuit a fixé l'entrée de son club à 12 euros. Mais il existe des cas particuliers : tous les 18-25 ans ont 20% de réduction, et toutes les filles ont 50% de réduction. Les filles de 18-25 ans cumulent les 2 réductions (c'est-à-dire qu'une fille de 18-25 ans bénéficie de 50% supplémentaires sur le prix déjà réduit de 20% : ceci ne fait pas 70% de remise !). Naturellement, un mineur ne peut pas rentrer dans la boîte, et les plus âgés paient plein tarif. Écrire une fonction `prix_entree : int -> char -> float` qui prend l'âge et le sexe d'un individu et qui retourne le prix d'entrée pour cet individu. Votre fonction fera une erreur si un mineur veut entrer dans la boîte.
2. Le propriétaire, pour attirer des clients, décide d'organiser une opération spéciale anniversaires. Dans ce cadre, tous les clients pouvant justifier être nés le jour de leur visite dans la boîte bénéficieront d'une réduction proportionnelle à leur âge : une personne de 27 ans aura 27% de remise, une personne de 30 ans aura 30% de remise, etc. Cette remise sera calculée sur le prix d'entrée habituel de la personne, i.e. sur le prix d'entrée calculé à la question précédente. On partira du principe qu'une personne ne peut pas avoir plus de 100 ans. Écrire la fonction `prix_entree_anniversaire : int -> char -> float` correspondante.

3 Des secondes

1. Écrire une fonction `conversion : int -> int -> int -> int` qui prend un horaire exprimé en heures, minutes et secondes et renvoie ce même horaire converti en secondes.
2. Écrire une fonction `temps_ecoule : int -> int -> int -> int -> int -> int -> int` qui prend 2 horaires exprimés en heures, minutes et secondes et renvoie le temps écoulé (en secondes) entre eux. L'ordre des horaires est indifférent (le plus petit ne sera pas forcément en premier).

4 Jeu de fléchettes

On considère la cible de fléchettes ci-dessous. Ses zones de tir sont définies par les points (2,0), (4,0), (6,0) et (8,0) et rapportent de de 100 à 0 points. Écrire une fonction `gain_flechette : float -> float -> int` qui calcule le gain du tir en fonction des coordonnées x et y du point d'impact.



5 Décomposition de jetés de dés

On considère un dé à 3 faces numérotées 1, 2 et 3. On souhaite connaître, étant donné un entier x, combien de jetés de chaque valeur il faut faire pour obtenir ce nombre (en additionnant les valeurs des jetés). On cherche la solution favorisant les plus grandes valeurs de jetés. Par exemple, 4 correspond à 1 jeté valant 3, 0 jetés valant 2 et 1 jeté valant 1.

— Écrire 3 fonctions `nb_3`, `nb_2` et `nb_1`, chacune de type `int -> int`, permettant, à partir du nombre total **initial**, de connaître le nombre de jetés valant 3, 2 et 1.

— Écrire une fonction `decompose_des : int -> string` qui renvoie la décomposition complète.