Feuille de travaux pratiques
Modules

L'objectif de ce TP est d'apprendre à utiliser les modules de la bibliotheque standard d'OCaml, notamment les modules Set (`https://caml.inria.fr/pub/docs/manual-ocaml/libref/Set.S.html`) et Map (`https://caml.inria.fr/pub/docs/manual-ocaml/libref/Map.S.html`) implantant respectivement des structures de données d'ensemble et de liste associative.

1. Écrire une fonction $compare\_hauteurs$ de type $hauteur \rightarrow hauteur \rightarrow int$ qui prend deux arguments $hauteur1$ et $hauteur2$ et retourne :
   — -1 si $hauteur1$ est plus basse que $hauteur2$;
   — 0 si $hauteur1$ est égale à $hauteur2$;
   — 1 si $hauteur1$ est plus haute que $hauteur2$.
   Cette fonction identifiera Mi dièse avec Fa et Si dièse avec Do.
   Pour définir une structure de donnée d'ensemble de hauteurs, il faut tout d'abord définir un module $HauteurOrdonnee$ de la manière suivante :

$$\begin{array}{l} module\ HauteurOrdonnee = \\ \quad struct \\ \quad\quad type\ t = hauteur \\ \quad\quad let\ compare = compare\_hauteur \\ \quad end \end{array}$$

   Ce module implémente la signature [1] $OrderedType$ qui fournit une fonction de comparaison pour un type t donné.
   La bibliothèque Set d'OCaml permet ensuite de définir un module $EnsembleHauteurs$ d'ensemble de hauteurs via l'utilisation du **foncteur** $Set.Make$. Un foncteur permet de définir un modules paramétré par un autre module. On utilise pour cela la syntaxe suivante :

$$module\ EnsembleHauteurs\ =\ Set.Make(HauteurOrdonnee)$$

2. Comprendre le type des fonctions $EnsembleHauteurs.empty$, $EnsembleHauteurs.add$ et $EnsembleHauteurs.iter$.
3. On considère le type $score$ permettant de représenter une partition simple, défini par :

$$type\ score = Score\ of\ hauteur\ list$$

1. La notion de signature en OCaml correspond à la notion d'interface que l'on retrouve par exemple en Java.

Écrire une fonction $ensemble\_hauteurs$ de type $score \rightarrow EnsembleHauteurs.t$ qui prend en argument $partition$ de type $score$ et retourne l'ensemble des notes qui apparaissent dans cette partition. On pourra utiliser les fonction $List.fold\_left$ ou $List.fold\_right$ et la fonction $EnsembleHauteurs.add$ pour cela.

4. Écrire une fonction $affiche r\_hauteurs$ qui affiche par ordre croissant toutes les hauteurs d'un ensemble de hauteurs.
5. Définir un module $MapHauteurs$ qui définit une table associative dont les clés sont des hauteurs, en utilisant le foncteur $Map.Make$ de la bibliothèque Map d'OCaml.

   Comprendre le type des fonctions $MapHauteurs.empty$, $MapHauteurs.mem$ et $MapHauteurs.add$.
6. Écrire une fonction $enumeration\_hauteurs$ de type $score \rightarrow int\ MapHauteurs.t$ qui retourne une liste associative représentant le nombre d'occurrences pour chaque hauteur apparaissant dans une partition de type $score$.
7. Écrire une fonction $affiche r\_map\_hauteurs$ de type $int\ MapHauteurs.t \rightarrow unit$ qui affiche par ordre croissant de la hauteur le nombre d'occurence de chaque hauteur d'un élément de type $MapHauteurs.t$.