1 Inférence de type

1. Traduire en Ocaml les expressions mathématiques suivantes et indiquer le type des expressions Ocaml obtenues.
   - $a \mapsto (n \mapsto n + a)$
   - $f \mapsto (x \mapsto f(1.5) * f(x))$
   - $f \mapsto (x \mapsto f(1) * f(x) * 1.5)$
   - $f \mapsto g \mapsto k \mapsto f(2 * k) *. g(3.5 *. float\_of\_int(k));;$
2. Indiquez si les expressions Ocaml suivantes sont valides, et si oui leur type et leur valeur :
   - `(2 + 4) <= (8 - 7);;`
   - `("a" ^ "b");;`
   - `(4 < 8) = ("a" = "b");;`
   - `("a" + 5) = ("a" + 5);;`
   - `function m -> m + 1;;`
   - `function m -> function n -> n +. m;;`
   - `function x -> (x   3) + 1;;`
   - `function x -> x  3;;`
   - `function b -> (f 0) or b;;`
   - `function b -> if a <= b then a else b;;`
   - `let som_fonc = function f -> function g -> function x -> f x  +.  g x;;`

2 Arrondis

2.1 Avec une précision fixe

1. Écrire une fonction Caml qui calcule l'arrondi au plus proche entier d'un décimal donné.
2. Écrire une fonction qui donne l'arrondi au plus proche demi-point (`arr 1.43 == 1.5`, et `arr 1.1 == 1`).

2.2 Avec une précision donnée

Écrire une fonction qui calcule des arrondis à n chiffres après la virgule, n étant un second paramètre de la fonction (arrondi 1.2345 2 == 1.23).

3 Type `char` et `string`

Fonction Ocaml	Type	Description
String.length	string -> int	Stringth ch = le nombre de caracères de ch
String.get	String->int->char	String.get ch n = le n-ème caractère de ch
String.sub	string -> int -> int -> string	String.sub ch d l = la souschaîne de ch de taille l, à partir du d-ème caractère

3.1 Type `char`

Écrire une fonction `est_voyelle` qui détermine si un caractère est une voyelle.

3.2 Des chiffres et des lettres

Écrire une fonction `verif_nf` qui accepte en argument une chaîne de caractères et un flottant et vérifie si les deux arguments représentent le même nombre.

3.3 Salutations

Écrire une fonction `bienvenu` qui prend en argument un booléen et deux chaînes de caractères pour le nom et le prénom et affiche un message d'accueil personnalisé tel que `Bonjour M. Jules Durand` ou `Bonjour Mme Ève Dupont`.

3.4 Problème de Conjugaison (A préparer pour le TP3)

Le but est d'écrire une fonction de conjugaison de verbes du premier groupe au futur. Pour cela on veut réaliser une fonction qui donne la conjugaison à une personne donnée d'une verbe donné.

Écrire une fonction `conjugue: bool -> int -> string -> string` qui prend en argument un booléen spécifiant le nombre (singulier(true),pluriel(false)), un entier pour le pronom personnel(compris entre 1 et 3) et une chîne de caractères pour le verbe. Le résultat de la fonction est la chaîne de caractères formée du bon pronom et du verbe conjugué au futur. Voici quelques exemples :

```
# conjugue true 2 "chanter" ;;(* verbe chanter à la 2eme personne du singulier *)
-:string = "tu chanteras"
# conjugue false 1 "manger" ;;
- : string = "nous mangerons"(* verbe manger à la 1ere personne du pluriel *)
```

En cas de paramètres non valides, la fonction devra déclencher une exception comme par exemple :

```
# conjugue false 5 "chanter";;
Exception: "Le 2eme paramètre doit \^etre compris entre 1 et 3"
# conjugue true 1 "dormir" ;;
Exception: "dormir n'est pas un verbe du premier groupe"
```

Pour résoudre ce problème en le décompose en sous-problèmes faciles à traiter. Une décomposition est proposée mais chacun est libre de résoudre le problème à sa manière.

1. Écrire une fonction `pronom: bool -> int -> string` qui prend en argument un booléen (singulier(false),pluriel(true)) et un entier (1 à 3) et retourne un pronom. Une exception sera déclenchée si i n'est pas valide. Par exemple

   ```
   #pronom true 1 ;;
   - : string = "je"
   ```

2. Écrire une fonction `premier_groupe : string -> bool` qui prend en argument une chaîne de caractères supposée être un verbe et vérifie si ce verbe se termine par "er" (utiliser String.get et String.length). Par exemple :

   ```
   #  premier_groupe "chanter";;
   - : bool = true
   ```

3. Écrire une fonction `terminaison: bool -> int -> string` qui retoune la terminaison des verbes en "er" au futur. Par exemple :

   ```
   #  terminaison true 3 ;;
   - : string = "a"
   ```

4. La fonction conjugue consistera à vérifier si le verbe est en "er" dans ce cas le résultat sera la concaténation du pronom,du verbe et de la terminaison, sinon une exception devra être déclenchée.