N'oubliez pas de réaliser plusieurs tests sur chaque fonction que vous définissez avant de passer àla suivante.

Exercice 1 Encore des fonctions ...

Question 1.1 évaluation statique
Taper les phrases suivantes

```
let a=3 and b=5;;
let f = function x-> a*x+b;;
f 4;;
```

Modifier les valeurs de a et b. Selon vous, quelle valeur devrait avoir f 4 ? Vérifier.

Question 1.2 Écrire une fonction `sommecar1` qui prend un entier n et calcule $\frac{n(n+1)(2n+1)}{6}$.

Question 1.3 Écrire une fonction `nb_rac` qui, étant donnés 3 flottants a, b, c, renvoie le nombre de solutions réelles de l'équation $ax^2 + bx + c = 0$. On évitera de calculer plusieurs fois le déterminant à l'aide d'une déclaration locale.

Exercice 2 Récursion

Question 2.1 Écrire une fonction `pgcd` qui calcule le pgcd de 2 entiers.

Question 2.2 Écrire une fonction `sommecar2` qui prend un entier n et calcule $\sum_{k=1}^{n} k^2$;
Puis écrire une fonction `test_sommecar` qui, pour n, vérifie que `sommmecar2 n` vaut $\frac{n(n+1)(2n+1)}{6}$ (refer 1.2).

Question 2.3 Écrire une fonction `sommeinv` qui calcule $\sum_{k=1}^{n} 1/k$.

Question 2.4 Écrire les fonctions u et v qui calculent les valeurs des suites u_n et v_n définies par

$$\begin{cases} u_n = 2v_{n-1} + 3, & u_0 = 4 \\ v_n = -u_{n-1} + 5, & v_0 = 0 \end{cases}$$

Question 2.5 Drôle de factorielle ... Pourquoi OCaml produit-il une erreur avec la phrase suivante

```
let factbiz n = if n=0 then 1 else n*factbiz(n-1);;
```

et pas avec

```
let factbiz = function n -> 5*n ;;
let factbiz n = if n=0 then 1 else n*factbiz(n-1);;
```

Qu'attendez-vous alors comme résultat de l'appel `factbiz 3` ? Vérifier.

Exercice 3 Filtrage

Question 3.1 La suite de Fibonacci est définie par $F_0 = 1$, $F_1 = 1$ et $F_n = F_{n-2} + F_{n-1}$ pour $n > 1$. Écrire une fonction qui calcule à partir d'un entier n, la valeur de F_n en utilisant une expression conditionnelle, puis à l'aide d'un filtrage.

Question 3.2 Implanter par filtrage la fonction d'Ackerman $ack : N \times N \to N$, définie par

$$\begin{aligned} ack(0,n) &= n+1 \\ ack(m,0) &= ack(m-1,1) \\ ack(m,n) &= ack(m-1, ack(m,n-1)). \end{aligned}$$

Question 3.3 Pour le filtrage de caractères, il existe une facilité qui permet de tester si un caractère appartient à un intervalle de caractères. En effet, une combinaison de motifs telle que `'e' | 'f' | 'g' | 'h'` peut s'écrire simplement `'e'..'h'`.

Écrire une fonction `discriminer` qui prend un caractère c et renvoie une chaîne "voyelle", "consonne", "chiffre" ou "autre" indiquant la nature de c.

Exercice 4 Fonctionnelles et récursion

Question 4.1 Écrire une fonction `concatene` qui, étant donnés une chaîne de caractères s et un entier n, retourne la chaîne formée de n fois la chaîne s (si $n = 0$, on renvoie la chaîne vide). Par exemple `concatene "hou" 3` renvoie "houhouhou".

On veut maintenant déclencher une exception par `failwith` lorsque $n < 0$. Faire en sorte de ne pas tester (inutilement) si n est négatif à chaque appel récursif : un seul test doit suffire.

Question 4.2 Écrire une fonction `compose` qui prend deux fonctions f et g et retourne la composée $f \circ g$ de ces deux fonctions.

Question 4.3 Écrire une fonction `itere_n` qui prend une fonction f et un entier $n \geq 0$, et renvoie la composée n fois de la fonction f, c'est-à-dire la fonction qui, à une valeur x associe $f^n(x)$. En donner une version qui utilise la fonction `compose` et une autre qui ne l'utilise pas.

Exercice 5 Des chiffres et des nombres
Soit n un entier naturel. On note $r_{k-1}r_{k-2}\ldots r_1r_0$ sa représentation en base b de sorte que

$$n = r_{k-1}b^{k-1} + \ldots + r_1b + r_0 \ .$$

On pose $q_0 = r_{k-1}b^{k-2} + \ldots + r_1$. On a donc

$$n = q_0b + r_0 \ .$$

Comment obtient-on en *Objective Caml* q_0 et r_0 à partir de n et b ?

Question 5.1 Écrire une fonction `somme_chiffres` qui prend un entier positif et calcule la somme de ses chiffres. À l'aide de la fonction prédéfinie `abs` (valeur absolue) et d'une déclaration locale, étendre la validité de `somme_chiffres` à tous les entiers.

Question 5.2 Déduire des relations ci-dessus un procédé de calcul du développement en base b de n (on supposera que $b \leq 10$) et écrire la fonction *Objective Caml* `baseb_of_10` correspondante.

Réciproquement, écrire une fonction `base10_of_b` qui prend b et un entier $r_{k-1}r_{k-2}\ldots r_1r_0$, et renvoie la valeur du nombre n (en base décimale) dont le développement en base b est $r_{k-1}r_{k-2}\ldots r_1r_0$.

Exercice 6 Manipuler les chaînes de caractères On peut concaténer 2 chaînes avec l'opérateur `^` directement, mais la plupart des fonctions disponibles se trouve dans le module `String`. Une fonction `fonc` de ce module est accessible par `String.fonc`; Elle peut cependant être appelée plus simplement par `fonc` à la condition qu'on ait ouvert auparavant le module par `open String`.

Consulter la description des fonctions du module String (standard library) dans le manuel de référence. Essayer les fonctions `length, get, index, sub` et s'habituer à la numérotation des caractères qui va de 0 à $l - 1$ pour une chaîne qui contient l caractères.

Question 6.1 Écrire une fonction `cherche_car` qui recherche si un caractère c apparaît dans une chaîne s : si oui, la fonction retourne la position de c dans s, et sinon le nombre -1.

Question 6.2 Étant donnée une chaîne s, on veut construire la chaîne s' en ajoutant un espace derrière chaque point et chaque virgule de s. Par exemple, si s vaut "dgf,aaf.gfd." alors s' vaut "dgf, aaf. gfd. ". Écrire une fonction `ajoute_espace` qui réalise ce travail.

Exercice 7 On veut créer une fonction générant, pour un entier $k > 0$, la suite de k lignes suivantes :
ligne 1 : 1
ligne 2 : 1 1
ligne 3 : 2 1
ligne 4 : 1 2 1 1
ligne 5 : 1 1 1 2 2 1
ligne 6 : 3 1 2 2 1 1
En fait, on obtient la ligne k en lisant ce qui est écrit sur la ligne $k-1$. Par exemple, la ligne 5 s'obtient en lisant la ligne 4 : sur cette dernière on voit 1 "1", 1 "2" et 2 "1".

1/ Écrivez une fonction `calc_prefixe` qui prend en argument une chaîne de caractères `str` ayant la forme des lignes mentionnées ci-dessus, et qui renvoie 0 si `str` est vide, ou le nombre de chiffres identiques au début de la chaîne `str`. Par exemple `calc_prefixe "1 1 1 2 2 1 "` renvoie la valeur 3 car la chaîne de caractères commence par trois "1".

2/ Écrivez une fonction `genere_ligne` qui prend en argument une chaîne de caractères correspondant à l'une des lignes décrites ci-dessus, et qui renvoie la ligne suivante. Pour simplifier, on supposera qu'après chaque nombre, il y a un espace. On admettra en outre que tous les nombres affichés sont compris entre 1 et 9.

3/ Écrivez une fonction `genere` qui prend en argument un entier $k > 0$, et qui renvoie une chaîne de caractères contenant les k premières lignes de la suite. La chaîne de caractères permettant d'aller à la ligne suivante est "\n".

4/ Écrivez une fonction `ecrit_suite` qui prend en argument un entier $k > 0$, et qui affiche la chaîne de caractères contenant les k premières lignes de la suite. La fonction permettant d'afficher une chaîne de caractères s'appelle `print_string`.