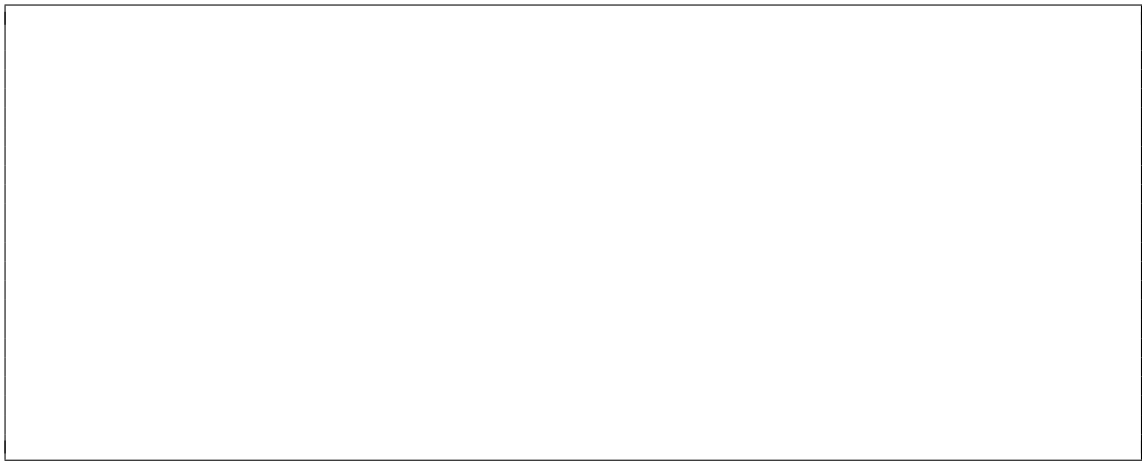
# Algorithmes et structures de données : Travail 2

Ce travail compte pour 6% de la session et porte sur la récursion. *50% des points seront retirés pour les implémentations non récursives*. Le code source pour le travail est disponible à `https://github.com/jfim/algorithmes-travail-2`. Vous ne pouvez pas utiliser les fonctions de la librairie standard. Vous devez remettre votre fichier `travail2.cpp` par LÉA avant le début du cours du 8 avril. La version papier du document doit être remise au début du cours du 8 avril.

Nom : _____

1. La série de Fibonacci est une série de chiffres importante qui apparait fréquemment dans la nature. Les deux premiers termes de celle-ci sont $f(0) = 0$ et $f(1) = 1$, puis les termes subséquents sont exprimés comme la somme des deux termes précédents, soit $f(n) = f(n-1) + f(n-2)$. Par exemple, $f(2) = f(1) + f(0)$, soit $f(2) = 1 + 0$.
   (a) (1 point) En utilisant la récursion, implémentez la fonction `fibonacci()`.
   (b) (½ point) Est-ce que la complexité de la version récursive est $O(n)$ ? ◯ Oui ◯ Non
   (c) (1 point) Dans vos mots, expliquez comment vous pourriez faire pour calculer cette série plus rapidement.



2. Le plus grand commun diviseur est le chiffre entier le plus grand qui divise deux chiffres de façon égale. Par exemple, le plus grand commun diviseur de 54 et 24 est 6.
   L'algorithme d'Euclide permet de trouver le plus grand common diviseur de deux nombres.

$$PGCD(n_1, n_2) = \begin{cases} n_1 & \text{si } n_2 = 0 \\ PGCD(n_2, n_1 \bmod n_2) & \text{si } n_2 > 0, \text{ où } \bmod \text{ est l'opérateur modulo} \end{cases}$$

   (a) (1 point) En utilisant la récursion, implémentez la fonction `pgcd()`.
   (b) (½ point) Est-ce que la complexité de cet algorithme est $O(n)$ ? ◯ Oui ◯ Non

3. La récursion terminale est un cas particulier de la récursivité où l'appel à la fonction récursive apparait comme la dernière instruction d'une fonction récursive. Par exemple, l'algorithme SOMME ci-dessous calcule la somme d'un tableau en utilisant la récursion terminale.

```
1: fonction SOMME(tableau, longueur, sommeAccumulee)
2:     si longueur = 0 donc
3:         retourne sommeAccumulee
4:     sinon
5:         retourne SOMME(tableau + 1, longueur − 1, sommeAccumulee + tableau[0])
6:     fin si
7: fin fonction
```

   (a) (1 point) Implémentez la fonction `maximum()` qui retourne le maximum d'un tableau de façon récursive.

   (b) ($\frac{1}{2}$ point) Est-ce que la complexité de cet algorithme est $O(n)$ ? ◯ Oui ◯ Non

   (c) ($\frac{1}{2}$ point) Selon vous, est-ce que cette version est plus rapide que la version qui utilise une boucle pour itérer sur les éléments du tableau ? ◯ Oui ◯ Non

   Expliquez pourquoi.

4. (2 points) La recherche binaire (ou dichotomique) permet de rechercher des éléments dans un tableau trié en un temps $O(\log n)$. Pour se faire, la recherche prend l'élément mitoyen puis le compare avec l'élément recherché. Si l'élément mitoyen n'est pas l'élément recherché, la recherche continue dans la moitié respective où l'élément recherché peut se trouver, tel qu'expliqué en classe.

   Implémentez une fonction `indexDe()` qui prend en paramètre un tableau, sa longueur et un chiffre à rechercher, puis qui retourne l'index du chiffre recherché en utilisant une recherche binaire implémentée avec la récursion. Dans le cas où le chiffre n'est pas dans le tableau, vous devez retourner l'index $-1$.

   Supposez que le tableau passé en paramètre est croissant. La fonction `indexDe()` n'a pas les paramètres que vous aurez besoin pour l'utiliser directement comme fonction récursive ; vous aurez donc à créer une autre fonction (qui sera récursive) avec les paramètres de votre choix et l'appeler à partir de `indexDe()`.