DS 1

Informatique pour tous, première année

Julien Reichert

La calculatrice n'est pas autorisée, et le détail des calculs doit apparaître sur la copie afin que tous les points soient accordés. En compensation, la durée de l'épreuve sera de 2 h pour laisser le temps de faire les calculs à la main.

Les nombres écrits ici n'ont pas la précision de la base dans leur représentation si elle est mentionnée juste avant dans les exercices demandant une conversion. Il faudra être attentif !

Exercice 1 : Calculer et donner le résultat en hexadécimal : $\overline{BABA}^{16} \times \overline{C881}^{16}$.

On signale pour l'exercice 2 que les nombres écrits en virgule flottante sur 16 bits ont cinq bits d'exposant.

Exercice 2 : Écrire le nombre $\frac{2019}{1024}$ en virgule flottante sur 16 bits.

Exercice 3 : Écrire une fonction `LC(n)` qui détermine si son argument est un nombre de Lucas-Carmichael.

On définit un nombre de Lucas-Carmichael comme un entier naturel n qui vérifie les propriétés suivantes :

- Le nombre n n'est pas premier.
- Pour tout diviseur premier p de n, $p+1$ divise $n+1$.
- Il n'existe pas de nombre p tel que p^2 divise n. Cette propriété est ajoutée par convention pour éviter de considérer comme des nombres de Lucas-Carmichael les nombres de la forme p^3+1 avec p premier. La vérification de la propriété peut ne se faire que pour les nombres premiers, la condition reste nécessaire et suffisante.

Pour résoudre l'exercice, on pourra commencer par écrire une fonction `diviseurs(n)`, listant les diviseurs du nombre en argument, éventuellement à l'exclusion des nombres 1 et `n`, ainsi qu'une fonction `carreparfait(d)`, déterminant si un nombre est un carré parfait dans le cadre de la vérification de la troisième propriété.

Exercice 4 : Écrire une fonction `couples(l, ll)` qui renvoie à partir de ses arguments une liste contenant les couples d'indices dans les listes respectives où un élément de `l` est égal à un élément de `ll`. L'ordre n'importe pas.

Par exemple, l'appel `couples(l, ll)` avec `l = [1, 2, 3, 3, 4]` et `ll = [3, 3, 1]` renverra avec un parcours naturel `[(0, 2), (2, 0), (2, 1), (3, 0), (3, 1)]`.

Exercice 5 : Écrire une fonction `supprimezone(l, x, debut, fin)` qui renvoie à partir de ses arguments une nouvelle liste qui est comme `l` mais dont le premier élément égal à `x` entre l'indice `debut` (inclus) et `fin` (exclu) est retiré. Si `x` n'est pas dans l'intervalle en question, la liste retournée est la même que `l`, et s'il y est plusieurs fois, c'est bien une seule occurrence qui est retirée.

Par exemple, l'appel `supprimezone([1, 2, 3, 4], 3, 0, 2)` renverra la liste en entrée, car l'indice où se trouve l'élément `3` n'est pas entre `0` inclus et `2` exclu, alors que l'appel `supprimezone([1, 2, 3, 4], 3, 2, 3)` renverra la liste `[1, 2, 4]`.