Interrogation du 30 septembre 2024

durée : 1 heure

Pour chacun des algorithmes, on justifiera avec soin :
- que l'algorithme termine (si une boucle « tant que » est utilisée).
- que l'algorithme renvoie bien le résultat demandé.

Une réponse non justifiée sera notée sur les trois quarts des points.

On peut utiliser les opérateurs `//` et `%` sur les entiers. On évitera d'utiliser des fonctionnalités trop avancées de python, surtout celles non vues en TD : on rédigera en pseudo-code avec des techniques élémentaires. En cas de doute, poser simplement la question.

Pour les boucles « pour », on adoptera les conventions suivantes. Si $a, b \in \mathbb{Z}$, « pour i allant de a à b » signifie « pour i parcourant en croissant l'intervalle $[\![a, b]\!]$ ». Lorsque $b < a$, cet intervalle est vide, donc aucune des instructions dans la boucle « pour » n'est effectuée (ce sera par exemple le cas si on écrit « pour i allant de 1 à n », avec $n = 0$). On n'utilisera pas d'autre type de boucles « pour » (décroissantes, saut d'indice etc) : dans ces situations, on utilisera l'instruction « tant que ».

Exercice 1. *Écrire un algorithme itératif, puis un autre récursif, recevant un entier n naturel en entrée, et retournant $n!$ en sortie.*

Exercice 2. *Écrire un algorithme itératif, puis un autre récursif, recevant une liste non vide de nombres en entrée, et retournant la valeur minimale dans cette liste.*

Exercice 3. *Écrire un algorithme itératif, puis un autre récursif, recevant un entier naturel et retournant la liste de ses chiffres en base* 10 *(dans l'ordre usuel : le chiffre des unités en dernier).*

Dans la suite, on utilise des chaînes de caractères. On les manipule comme des listes, en particulier la concaténation sera notée « + », et la longueur s'obtient avec une fonction `longueur` que l'on peut librement utiliser. Les différentes lettres d'une chaîne de caractères sont accessibles par leur indice comme une liste : $c[0]$, $c[1]$ etc. Une chaîne de caractères **peut être vide**, auquel cas elle est de longueur zéro.

Exercice 4. *On dit qu'une chaîne de caractères s est un carré s'il existe une chaîne de caractères a telle que s soit égale à a concaténée avec elle-même. Par exemple,* `'papa'` *est un carré, mais pas* `'maman'`.

1. *Écrire une fonction* `est_un_carre(c)` *qui prend en entrée une chaîne de caractères et qui retourne* `True` *si c'est un carré et* `False` *sinon.*
2. *Si la chaîne en entrée est de longueur n, majorer le nombre de comparaisons entre lettres effectuées par la fonction dans le pire des cas.*

Exercice 5. *On dit qu'une chaîne de caractères s contient un carré si c'est la concaténation de trois chaînes de caractères a, b et c (dans cet ordre) et que b est un carré* ***non vide****. Par exemple,* `'maman'` *contient un carré (prendre a la chaîne vide, b la chaîne* `'mama'` *qui est bien un carré et c la chaîne* `'n'`*).*

1. *Écrire une fonction* `contient_un_carre(s)` *qui prend en entrée une chaîne de caractères et qui retourne* `True` *si elle contient un carré et* `False` *sinon. On pourra utiliser la fonction* `est_un_carre` *de l'exercice précédent.*
2. *Si la chaîne en entrée est de longueur n, majorer le nombre de comparaisons entre lettres effectuées par la fonction dans le pire des cas.*

Exercice 6. *Un triplet pythagoricien ordonné est un triplet d'entiers naturels non nuls* (a, b, c) *vérifiant* $a \leq b \leq c$ *et* $a^2 + b^2 = c^2$.

Écrire une fonction `triplets_pyathgoriciens(N)` *prenant un entier* N *naturel en entrée et retournant la liste de tous les triplets pythagoriciens ordonnés* (a, b, c) *avec* $c \leq N$. *Chaque triplet pythagoricien sera retourné sous forme de liste, donc la fonction doit retourner une liste de listes. Par exemple,* `triplets_pyathgoriciens(2)` *doit retourner* [], `triplets_pyathgoriciens(6)` *doit retourner* $[[3, 4, 5]]$ *(liste contenant un unique triplet pythagoricien ordonné), et* `triplets_pyathgoriciens(13)` *doit retourner une liste contenant trois éléments, ces éléments devant être* $[3, 4, 5]$, $[6, 8, 10]$ *et* $[5, 12, 13]$ *dans un ordre que l'on n'impose pas.*