⚠ La calculatrice est interdite. Écrire l'ensemble des fonctions en langage `Python` en faisant attention à bien indenter.

On rappelle que l'opérateur `%` renvoie le reste de la division euclidienne, ainsi `5%3` renvoie `2`. Lorsque le sujet demande d'utiliser des piles (au lieu de tableaux/listes), on pourra utiliser les 6 fonctions suivantes :

fonction	`creer_pile()`	`empiler(p,e)`	`depiler(p)`	`taille(p)`	`est_vide(p)`	`sommet(p)`
variables	$\emptyset$	p : pile e : element	p : pile	p : pile	p : pile	p : pile
sortie	une pile	$\emptyset$	un élement	un entier	un booléen	un élement

Exercice 1

On considère la suite définie par $u_0 = 1$ et $u_n = 6u_{n-1}^3 - 1$, $\forall n \in \mathbb{N}^*$.

a) À l'aide d'une boucle `for`, écrire un programme qui affiche successivement les 50 premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$.

b) À l'aide d'une boucle `while`, écrire un programme qui renvoie l'indice et le premier terme de la suite $(u_n)_{n\in\mathbb{N}}$ supérieur ou égale à 500.

Exercice 2

On dit qu'un entier naturel n est premier si, et seulement si, il admet exactement deux diviseurs : 1 et lui-même. 0 et 1 ne sont donc pas des nombres premiers. Par contre, 3 est un nombre premier puisque l'ensemble de ses diviseurs est exactement $\{1;3\}$.

On pourra au fil des questions utiliser les fonctions construites dans les questions précédentes.

a) Écrire une fonction `divise(p,q)` d'argument deux entiers naturels non nuls p et q, renvoyant `True` si p divise q et `False` sinon.

b) Écrire une fonction `estpremier(p)` d'argument un entier naturel p, renvoyant 1 si p est premier et 0 sinon.

c) Écrire une fonction `nextprime(N)` qui prend un argument entier N et qui retourne comme valeur le premier nombre premier qui est strictement supérieur à N.

d) Écrire une fonction `phi(p)` d'argument un entier naturel p et renvoyant le nombre de nombres premiers inférieurs ou égaux à p.

e) D'après le théorème des nombres premiers, $\phi(n) \underset{n\to+\infty}{\sim} \dfrac{n}{\ln(n)}$.

Ainsi, si on définit pour tout $n \in \mathbb{N}^*$, $\Theta(n) = \left|\dfrac{\phi(n)\ln(n)}{n} - 1\right|$, on a $\lim\limits_{n\to+\infty} \Theta(n) = 0$.

Écrire une fonction `test(epsilon)` d'argument un réel epsilon strictement positif, renvoyant le premier entier naturel $N \geqslant 50$ tel que $\Theta(N) \leqslant epsilon$.

Exercice 3 (avec des piles) :

Pour les deux questions suivantes, on considère des piles ne contenant que des nombres flottants.

a) Écrire une fonction `ordreCroissant(p)` qui prend en argument une pile `p` et renvoie `True` si la pile est ordonnée dans l'ordre croissant, `False` sinon (⚠ la base devra être le plus petit élément).

b) Écrire une fonction `insertion(p, x)` qui prend en argument une pile `p` ordonnée dans l'ordre croissant et un nombre `x`, et qui renvoie une nouvelle pile toujours ordonnée dans laquelle le nombre x a été inséré.

Les piles contiennent maintenant des caractères. On dit qu'une expression est un palindromme s'il possède un axe de symétrie (seules les lettres comptent ; ni les accents ni les espaces ne comptent).
Ainsi "Abba", "121", "Un radar nu", "Élu par cette crapule" sont des palindrommes, mais "Bonjour", "Coucou", "1313", "Elle alla à l'aile" n'en sont pas. Par la suite, on suppose que les expressions fournies ne contiennent aucun accent, majuscule ni espace.

c) Écrire une fonction `convertStr2piles(phrase)` qui prend en argmuent une chaîne de caractères et qui renvoie une pile composée de chacun des caractères dans l'ordre.
expl : `convertStr2piles("cpge")` renvoie une pile de hauteur 4 dont le sommet est "e")

d) Écrire une fonction `VerifPalindromme(p)` qui prend en argument une pile et renvoie un booléen : `True` si la phrase contenue dans la pile est un palindromme, `False` sinon.