TP : nombres premiers
Informatique commune

1. Écrire une fonction `diviseurs` renvoyant la liste des diviseurs positifs d'un entier n. Par exemple, `diviseurs(36)` doit renvoyer `[1, 2, 3, 4, 6, 9, 12, 18, 36]`.

2. En déduire une fonction `premier` déterminant si un entier est premier.
3. En déduire une fonction `tous_premiers` telle que `tous_premiers(n)` renvoie la liste des nombres premiers inférieurs à n. Quelle est sa complexité ?

Le crible d'Ératosthène est un algorithme plus efficace pour obtenir la liste des nombres premiers inférieurs à un entier n :
On commencer par créer une liste `L` de taille $n+1$ dont tous les éléments sont `True`. On modifie la valeur de `L[0]` et `L[1]` à `False`. Puis pour chaque indice i de `L`, si `L[i]` contient `True`, alors pour chaque k multiple de i, on modifie `L[k]` en `False`. À la fin, `L[i]` vaut `True` si et seulement si i est premier.
Si vous ne comprenez pas, vous pouvez chercher plus d'explications sur Google (activité de 3ème par exemple).

4. Appliquer à la main l'algorithme d'Ératosthène pour $n = 100$, en barrant les nombres quand ils sont mis à `False` (les nombres non barrés à la fin sont les nombres premiers) :

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

5. Écrire une fonction `eratosthene` telle que `eratosthene(n)` renvoie la liste `L` ci-dessus.
   Vérifier que votre fonction marche avec l'exemple de la question précédente.
6. On veut déterminer expérimentalement la complexité de l'algorithme d'Ératosthène. Pour cela, créer une variable pour compter le nombre de fois que `eratosthene` met une valeur de `L` à `False` puis l'afficher juste avant le `return`. Comparer avec la complexité de `tous_premiers`.
7. Écrire une fonction `multiplicite` telle que `multiplicite(d, n)` renvoie le plus grand entier k tel que d^k divise n.
8. Écrire une fonction `decomposition` ayant un argument `n` et qui renvoie la liste `L` des diviseurs premiers de `n` avec multiplicités. On pourra stocker dans chaque élément de `L` une liste composée de deux entiers naturels : le diviseur et sa multiplicité. Par exemple `decomposition(50)` devra renvoyer la liste `[[2, 1], [5, 2]]`, puisque $50 = 2^1 \times 5^2$.