1. Pour tout entier naturel n, on appelle $\pi_1(n)$ le nombre de nombres premiers p vérifiant à la fois $p \leq n$ et $p-1$ est multiple de 4. De même, on appelle $\pi_3(n)$ le nombre de nombres premiers p vérifiant à la fois $p \leq n$ et $p-3$ est multiple de 4.
   (a) Écrire la fonction `pi1[n_]:= Module[{...},...]` (Documentation : `Do`, `PrimeQ`, `Mod`).
   (b) Faire de même pour `pi3[n_]`.
   (c) Vérifier la pertinence de vos fonctions : par exemple, $\pi_1(12345) = 727$ et $\pi_3(12345) = 746$.
   (d) Afficher la liste des entiers $\pi_3(k) - \pi_1(k)$ pour k variant entre 1 et n, en prenant successivement pour valeurs de n : 100, 200, 400, 800, 1600 (Documentation : `Table`).
   (e) Quel est le temps mis pour réaliser chacun des calculs ci-dessus (Documentation : `Timing`) ? Quand on double la valeur de n, que fait le temps de calcul ? Quel serait le temps approximatif de calcul pour $n = 50000$?
   (f) On remarque que toutes les quantités obtenues sont positives, et on serait tenté de conjecturer que $\pi_3(n) \geq \pi_1(n)$ pour tout entier n. En fait, il n'en est rien.
   Nous désirons maintenant déterminer le plus petit entier n tel que $\pi_1(n) > \pi_3(n)$ mais il serait relativement téméraire d'utiliser les fonctions déjà créées : il faut écrire les choses différemment.
   Créer une fonction `inversion[n_]` construite de la façon suivante : cette fonction comporte deux variables locales s_1 et s_3 initialisées à 0, et une variable locale p initialisée à 2. Tant que $s_1 \leq s_3$ et $p \leq n$ (Doc : `While`), on ajoute 1 à p. Si p est premier, alors on ajoute 1 à s_1 ou à s_3 (à vous de décider). Une fois la boucle terminée, on renvoie `ECHEC` si $p > n$, et la liste $\{p, s_1, s_3\}$ dans le cas contraire.
   Une fois cette fonction créée, déterminer le plus petit entier n tel que $\pi_1(n) > \pi_3(n)$.
2. Le *petit théorème de Fermat* dit que pour tout nombre premier p et tout entier a, le nombre p divise $a^p - a$.
   Étant donné un entier naturel $m \geq 2$ et un entier a , on dira que a est un témoin de Fermat pour m lorsque m divise $a^m - a$. Ainsi, le petit théorème de Fermat nous dit que si un nombre p est premier, tous les entiers sont des témoins de Fermat pour p.
   (a) Écrire une fonction `temoin[a_,m_]` qui renvoie `True` si a est témoin de Fermat pour l'entier m, et `False` sinon (Doc : `PowerMod`).
   (b) Écrire une fonction `listeTemoins[a_,n_]` renvoyant la liste de tous les nombres *non premiers* inférieurs à n pour lesquels a est un témoin de Fermat (Doc : `Not`, `Append`).
   (c) Calculer les listes s_2, s_3, s_5, s_7, s_{11} des nombres non premiers inférieurs à 100000 pour lesquels 2, 3, 5, 7, 11 sont témoins de Fermat.
   (d) Déterminer l'intersection de ces listes.
   (e) Rechercher sur Internet ce qu'est un *nombre de Carmichael*, ainsi que la liste des nombres de Carmichael inférieurs à 100000. Conclusion ?
3. Un célèbre théorème dit que pour tout nombre premier p impair, le nombre p divise $2^{p-1} - 1$. Mais existe-t-il des nombres premiers p tels que p^2 divise $2^{p-1} - 1$? Un tel nombre p est appelé un nombre de Wieferich. Trouver deux nombres de Wieferich (ils sont inférieurs à 10000). Vous pouvez éventuellement essayer d'en chercher un troisième, mais sachez-le, *il n'y a que deux nombres inférieurs à* 10^{15} *qui sont solution du problème.*
4. Pour tout entier naturel n, le n-ième nombre de Mersenne est le nombre $M_n = 2^n - 1$. On montrera en TD que si ce nombre est premier, alors n est forcément premier. En revanche, la réciproque est fausse. Mais il existe un test très efficace, appelé *test de Lucas* pour savoir (lorsque n est premier) si M_n est premier. On pose $s_1 = 4$, puis, pour $k = 2, \ldots, n-1$, on pose $s_k = s_{k-1}^2 - 2 \mod m$ (où $m = 2^n - 1$). Alors, m est premier si et seulement si s_{n-1} est divisible par m.
   (a) Écrire une fonction `testLucas[n_]` qui renvoie `True` si et seulement si le nombre M_n est premier.
   (b) Trouver tous les nombres de Mersenne M_n premiers pour $n \leq 2012$ (Doc : `Select`, `Map`).