Ce code en C implémente le test de primalité de Fermat pour déterminer si un nombre est probablement premier. Voici un compte rendu détaillé des choix effectués pour chaque fonction :

I. GenerateRandomNumber:

Cette fonction génère un nombre aléatoire de 2048 bits en utilisant la bibliothèque GMP (GNU Multiple Precision Arithmetic Library). Elle utilise également la fonction gettimeofday pour obtenir un seed basé sur le temps actuel en millisecondes. Ensuite, elle initialise un état de générateur de nombres aléatoires, utilise cette seed pour initialiser l'état, et génère un nombre aléatoire de 2048 bits avec mpz_urandomb. Enfin, elle nettoie l'état du générateur de nombres aléatoires. Les nombres générés seront utilisés pour les tests de primalité.

II. mpz_expo_mod:

Cette fonction calcule l'exponentiation modulaire en utilisant l'algorithme de l'exponentiation rapide (square and multiply) avec le modulo pour éviter les calculs exponentiels coûteux. Elle prend en entrée trois mpz_t (nombres GMP) : la base, l'exposant et le modulo, et calcule (base^exp) % mod. Pour cela, elle initialise deux mpz_t copies de la base et de l'exposant, puis itère sur les bits de l'exposant pour effectuer les calculs de manière efficace en utilisant les opérations modulaires. Elle nettoie ensuite les variables temporaires.

III. FermatTest:

Algorithme 6 : Test de primalité de Fermat

Données : Un entier n et un nombre de répétitions k
pour $i = 1$ *jusqu'à* k **faire**
  Choisir aléatoirement a tel que $1 < a < n - 1$;
  si $a^{n-1} \neq 1 \mod n$ **alors**
    renvoyer `composé` ;
renvoyer `premier` ;

Cette fonction implémente le test de primalité de Fermat pour déterminer si un nombre est probablement premier. Elle prend en entrée le nombre à tester (n) et le nombre d'itérations (k) pour le test. Elle initialise un générateur de nombres aléatoires avec un seed basé sur l'heure actuelle, puis effectue k itérations du test. À chaque itération, elle génère un nombre aléatoire 'a' dans l'intervalle [1, n-1] et calculer a^(n-1) % n. Si le résultat n'est pas égal à 1, alors n est composé. Sinon, elle répète le processus pour un autre nombre aléatoire. Si toutes les itérations

passent, alors le nombre est probablement premier sinon il y a une forte probabilité qu'il soit composé.

Il faut noter qu'il peut y avoir des faux positifs aussi connus sous le nom de Fermat's Liar, ceux-ci sont des nombres de Carmichael. Par faux positif on désigne le cas où un nombre composé valide la propriété suivante : $a^{n-1} \equiv 1 \pmod{n}$

En pratique on utilise le test de Fermat en conjonction avec des test de division par des premiers connus et suivant le résultat de ces test on va appliquer d'autres algorithmes comme Miller-Rabin.

IV. Main:

La fonction principale du programme. Elle itère 10 fois pour générer 10 nombres aléatoires et tester leur primalité en utilisant le test de Fermat. Elle imprime ensuite les nombres générés et les résultats des tests.

En résumé, ce code utilise la bibliothèque GMP pour manipuler des nombres entiers de grande taille, implémente le test de primalité de Fermat pour tester la primalité des nombres générés aléatoirement, et utilise des techniques efficaces pour les calculs exponentiels modulaires afin d'améliorer les performances du programme.

V. Sources

- https://www.bibmath.net/dossiers/index.php?action=affiche&quoi=dossier1/page3.html

- https://en.wikipedia.org/wiki/Fermat_primality_test