Ce projet est réalisé dans le cadre d'un DM du module cryptographie, dans le but d'implémenter en C un test de primalité qui est : le test de Solovay-Strassen.Ce test emploie ce qu'on appelle le symbole de Jacobi pour tester si un nombre est premiers ou pas.
Pour faire ce test nous avons eu besoins du calcul de symbole de jacobi en utilisant toute les propriétées de ce dernier ainsi que L'algorithme de Square and Multiply , Test de parité, Test de congruence modulo … etc

Le projet contient 6 fichiers :

1. aritmethic.c et son .h: *contiennent Le test de parité, le test de congruence modulo et le teste de si deux nombres sont premiers entre eux.*
2.square_and_multiply.c et son .h: *contiennent l'algorithme d'expenentiation modulaire.*
3.symbol_of_jacobi.c et son .h : *contiennent toutes les propriétées du symbole de jacobi et l'algorithme qui calcule ce symbole en utilisant toutes ces propriétées.*
4.solovay_strassen.c et son .h: *contiennent l'algorithme du "Test de primalité de Solovay-Strassen".*
5.symbol_of_legendre.c et son .h: *contiennent Le test de residu quadratique et le calcul du sybole de legendre.*
6.main.c

Les Algorithmes utilisés :

`_Bool is_even_or_odd(mpz_t x):`

Cette fonction se charge de tester la parité du nombre passé en paramètre, en testant si le reste de la division du nombre est égal a 1 (impair) ou 0 (pair) et retourne True s'il est pair et False sinon.

`_Bool is_congruent(mpz_t n, unsigned long int mod, unsigned long int val):`

Cette fonction teste la congruence de deux nombre modulo *mod et renvoie true s'ils sont congruents False sinon.*

`_Bool check_coprimes(mpz_t x, mpz_t y) :`

Cette fonction teste si deux nombres sont **premiers** entre eux en calculant d'abord le pgcd des deux nombre *x* et *t* à l'aide de l'**ALGORITHME D'EUCLIDE.** Cela revient à faire une division euclidienne en traitant tout les cas de figures.
Cette fonction retourne TRUE si le pgcd est 1 et donc les deux nombres sont premiers entre eux sinon FALSE.

`void sq_and_mul(mpz_t result,mpz_t x,mpz_t a,mpz_t p)` :

Cette fonction se charge de **l'exponentiation modulaire** rapide (SQUARE AND MULTIPLY):

Tout d'abord,On converti L'exposant a en binaire (en faisant des divisions successives du nombre sur 2). Ensuite, on lit le resultat de la conversion Caractère par caractère et on s'arrete a l'avant dernier Bit (selon l'algorithme de Square and multiply on ne lit pas le dernier bit du poids forts). Et on teste a chaque fois si le bit == 0 on fait un SQUARE sinon (bit == 1) on fait un SQUARE and MULTIPLY. On prend a chaque itération le reste de la division du resultat sur le Modulo p

Pour finir on stock le résultat final dans la variable result.

`int first_proprety(mpz_t temp, mpz_t a, mpz_t n) :`

cette Fonction réalise la **propriété 1** du symbole de jacobi de (a n). Prend en Arguments : a un entier , n le nombre qu'on cherche à savoir s'il est premier ou pas. Retourne 1 si on peut réduire a modolu n, 0 sinon.

`int second_proprety(mpz_t a, mpz_t n)` :

Cette fonction réalise la **propriété 2** du symbole de jacobi (a ,n)prend en Entrée un entier a ,et n le nombre qu'on cherche à savoir s'il est premier ou pas.Retourne 0 si n divise a et la valeur du symbole_de jacobi sinon.

`int third_proprety(mpz_t a, mpz_t value, mpz_t pow):`

Une fonction qui réalise la **propriété 3** du symbole de jacobi

Prends a en Entrée,un entier à décomposer si possible en facteurs et stock les facteurs dans value et pow. Retourne 1 si a est un multiple de deux 0 sinon.

`int fourth_proprety(mpz_t a):`

Une fonction qui réalise la **propriété 4** du symbole de jacobi, prend une entier a comme argument et retourne 1 si a = 1 et donc jacobi(a,n) = 1 et 0 sinon.

`int fifth_proprety(mpz_t n)`:

Une fonction qui réalise la **propriété 5** du symbole de jacobi, elle a n comme Argument (le nombre qu'on cherche à savoir s'il est premier ou pas). Et retourne 1 si n = 1 mod 8 ou n = 7 mod 8, -1 si n = 3 mod 8 ou n = 5 mod 8 et 0 sinon.

`int sixth_proprety(mpz_t m, mpz_t n):`

Une fonction qui réalise la **propriété 6** du symbole de jacobi (Loi de réciprocité quadratique)ses Arguments sont : n le nombre qu'on cherche à savoir s'il est premier ou pas, m un entier.

Cette fonction retourne 1 si m et n sont premier et que n ou m = 1 mod 4, et -1 si n et m = 3 mod 4 ou n = 5 mod 8 et 0 sinon.

Tout ces propriétés vont servir a calculer le symbole de jacobi (a n):

`int symbol_of_jacobi(mpz_t a, mpz_t n)`

C’est la fonction qui calcule **le symbol de jacobi** selon l'algorithme de notre Professeur.en faisant des appels récursives entre les propriétes présenté précedemment en respectant l’ordre des 4 étapes de l’algorithme.

Elle prend comme argument a un entier , n un nombre entier dont on souhaite tester la primalité il ne doit être ni égal à 0 ni pair.

Cette fonction retourne **1** si a = 1 et donc jacobi(a,n) = 1 et **0** sinon.

`_Bool solovay_strassen(mpz_t n, unsigned long long int k) :`

Cette fonction réalise **le teste de primalité de solovay strassen** en partant du symbole de jacobi et en utilisant la méthode de l'exponentiation modulaire pour accélerer le calcul de grandes puissances. Ses Arguments sont : a un entier , n le nombre à tester et elle retourne **TRUE** s'il est probablement premier et **FALSE** s'il est composé.

`_Bool is_quadratic_residu(mpz_t a, mpz_t p)` :

Une fonction booléenne qui vérifie si un nombre n est **résidu quadratique** modulo un autre nombre p.

`void symbol_of_legendre(mpz_t a, mpz_t p) :`

cette fonction calcule le symbole de legendre en utilisant le test de Primalité de solovay-strassen et l'algorithme d'expenentiation modulaire selon le critère d'Eulere.

Compilation et Execution :

un makefile a été créé avec dedans plusieurs cibles,les cibles gmp et uvsqgraphics permettent d'installer les deux bibliothèques.

Afin de compiler le programme il faudra taper make dans le terminal.Ensuite il y'a 4 possibilités d'execution :

1- ./solovay_strassen primality_test n : afin de réaliser le test de solovay strassen.

2- ./solovay_strassen symbol_of_jacobi a n: afin de calculer le symbole de jacobi (a/n)

3- ./solovay_strassen symbol_of_legendre a n: afin de calculer le symbole de Legendre (a/n)

4- ./solovay_strassen coprimes a n : afin de tester si les deux nombres sont premiers entre eux .