Assignment 6: Public Key Cryptography
This program generates a pair of RSA keys, one public and one private. The public key can then be used by the program to encrypt a user selected file. The private key can be used to decrypt a file that has been encrypted by any encryption program that used the pregenerated public key.

encrypt.c:
One of the three main files that can be run to encrypt a user inputted file through the use of the public RSA key gene rated by the keygen
file program.
Can be launched with any of 5 different command options.
-i: specifies the input file to encrypt
-o: specifies the outfile file to print the encrypted file
-n: specifies the file containing the public key
-v: enables verbose output
-h: displays program synopsis and usage
Launching with verbose should result in the username, the signature, the public modulus and the public exponent bei ng printed.
Unlike asgn5, reading and writing from the user inputted files can be done via gmp functions, as a result the logic of both encrypt and
decrypt are rather simple. Most of the work falls to the rest of the files.

decrypt.c:
One of the three main files that can be run to decrypt a user inputted file through the use of the private RSA key gene rated by the keygen
file program.
Can be launched with the exact same command options as encrypt.c. A key exception being verbose resulting in onl y the public modulus and
the private key being printed alongside the decrypted output. Just like encrypt, most of the work of decrypt falls to th e other files
and gmp.

keygen.c:
One of the three main files that can be run to generate a new RSA key pair.
Can be launched with any of 5 different comman options.
-b: specifies the min bits needed for the public modulus
-i: specifies the number of Miller-Rabin iterations for testing primes
-n: specifies the public key file
-d: specifies the private key file
-s: specifies the random seed for the random state initialization
-v: enables verbose output
-h: displays program synopsis and usage
Akin to encrypt and decrypt most of the work of actually generating an RSA key pair falls on gmp and the following files to imitate gmp.

numtheory.c:
The function of the functions within this file are not terribly difficult to understand. The main difficult of this file co mes from the
fact that all functions within must be manually created via far more efficient algorithms than would likely be done ot herwise.
pow_mod - calculates the mod of an exponentiated integer divided by the requested modulus number
is_prime - calculates whether the specified integer is prime, but must be done via the Miller-Rabin primality test
make_prime - calculates a random prime number that is at minimum a specified number of bits and with a requested number of iterations
gcd - calculates the greatest common denominator of the two specified variables through the use of Euclid's alogorit hm

mod_inverse - calculates the modulus inverse via Bezout's identity

randstate.c:
Quite possibly the shortest of all of the files in this assignment, randstate does 2 things. First it allows the user to set a custom seed
that is used to generate all random numbers. Secondly it can clear all memory used by the state by simply making a s imple call
to gmp_randclear().

rsa.c:
The real meat and potatoes when it comes to actually generating RSA keys and encrypting/decrypting files.
Comes with a large number of functions that are used within all three of the main files.
rsa_make_pub - creates the parts p,q,n and e of a public RSA key
rsa_write_pub - prints a given rsa public key to the requested output file, printed in hexstrings
rsa_read_pub - reads a rsa public key from a given input file in the form of n,e,s and the username. n,e and s being in hexstrings
rsa_make_priv - creates a new rsa private key given p,q and e from rsa_make_pub
rsa_write_prive - prints a given rsa private key to the requested output file
rsa_read_priv - reads an rsa private key from a given input file
rsa_encrypt - performs rsa encryption given c,m,e and n
rsa_encrypt_file - encrypts the given input file and writes the encrypted version to the given output file.
The file must be encypted in blocks of size less than n.
rsa_decrypt - performs rsa decryption given m,c,d and n
rsa_decrypt_file - decrypts the given input file and writes the decrypted version the given output file
rsa_sign - generates an rsa signature using the private and public keys
rsa_verify - verifies the validity of an rsa signature