Assignment 3

Advanced Algorithms 1 (7081) --- Fall 2020

Due Wednesday, October 14 before midnight

REMINDER. Test 1 will be held during class time on Wednesday, October 7

*The **leader** of each group is to upload a .cpp file with the **source code** for your* C++ *program as well as a file with **output for a sample run**. For ease of grading, all the C++ code for your program should be included in a **single** file and designed using the C++ Visual Studio Platform. It should be well-commented and the output user-friendly.*

Textbook reference. RSA discussed in Chapter 1 and the Miller-Rabin primality testing algorithm discussed in Chapter 5 of the textbook *Algorithms: Special Topics*.

Topics covered: *RSA, modular exponentiation, changing bases, GCD, extended Euclid GCD, Miller-Rabin prime-testing algorithm.*

Write a C++ program that involves implementing the RSA cryptosystem. In practice for the encryption to be secure and to handle larger messages you would need to utilize a class for large integers. However, for this assignment you can use built-in types to store integers, e.g., `unsigned long long int`.

Also, rather than using the ASCII table for this assignment use BEARCATII, which restricts the characters to the blank character and the lower-case letters of the alphabet as follows:

blank character is assigned the value 0.
A, …, Z are assigned the values 1, …, 26, respectively.

The message M will be represented by replacing each character in the message with its assigned integer base 27. For example, the message M = "TEST" will be represented as

N = 20 5 19 20

Translating this to decimal we obtain:

$$D = 20 + 19*27 + 5*27^2 + 20*27^3 = 397838$$

Note that to convert back to base 27, we simply apply the algorithm we discussed in class, i.e., the least significant digit (rightmost) is obtained by performing the operations D mod 27 and performing a recursive call with D/27. For the example above we obtain,

397838 / 27, 397838 mod 27 = 14734, 20
→ 14734 / 27, 14734 mod 27, 20 = 545, 19, 20
→ 545/27, 545 mod 20, 19, 20 = 20, 5, 19, 20 = N

Find primes p and q by choosing positive integers at random and testing for primality using **Miller-Rabin** probabilistic algorithm.

Your program should prompt the user to input a positive integer representing the public key e. If the user enters a number that is not relatively prime to $n = pq$, then have the user reenter and keep doing this until e and n are coprime, i.e., $\gcd(e,\varphi(n)) = 1$. Also prompt the user to enter the message M (as a character string). For handing purposes, run your program with M = "TEST". Output p, q, n, M, C, P where C is the encrypted message, i.e., cyber text, and P is the decrypted message, i.e., plaintext. If your program is working correctly then M should be equal to P.