CSC281: Discrete Math for Computer Science
Computer Science Department
King Saud University

First Semester 1441
Project
Due Date: Sunday, November 10, 2019, 11:30 PM

This homework is designed to give you a hands-on understanding of the importance and applicability of discrete mathematics in computer science.

- ***You may work on this homework in pairs or group of three students.***
- ***Submission will be made by one student on behalf of the team on LMS.***
- ***Make sure to include the names of all members on the cover page.***
- ***All submitted code will be automatically checked for similarity, and if plagiarism is confirmed penalties, both groups will get ZERO in the project. The same criteria if somebody write you the code we will figure out that during project discussion.***
- ***The project discussion schedule will be determined later and it will be on the tutorial class.***

[RSA Cryptography]

Public key cryptosystems known as the **RSA system.** RSA cryptosystem is based on the dramatic difference between the ease of finding large primes and the difficulty of factoring the product of two large prime numbers (the integer factorization problem).

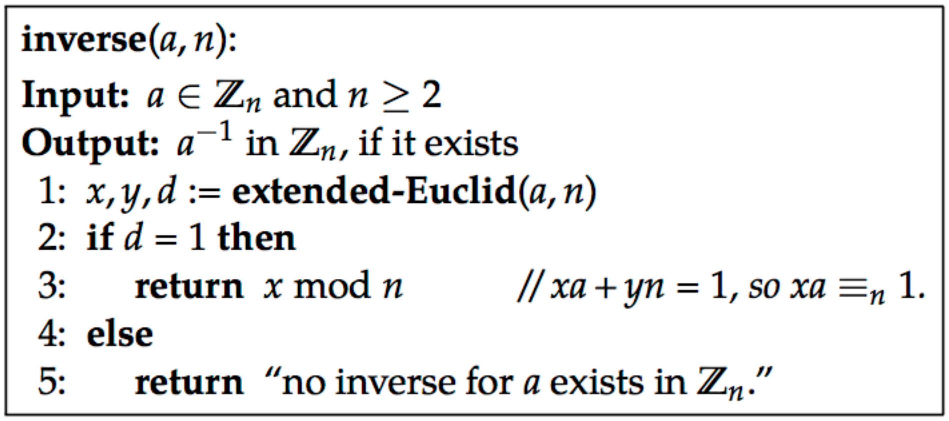
In this project, you have to write an implementation for the RSA cryptosystem in a console mode of Java. Your code should implement the following components:

- **`extended-Euclid`**`(int p, int q)`
  Finds the greatest common divisor of two integers p and q using the extended Euclid's algorithm as below:

```
extended-Euclid($n, m$):
Input: positive integers $n$ and $m \geq n$.
Output: $x, y, r \in \mathbb{Z}$ where $\gcd(n, m) = r = xn + ym$
1: if $m \bmod n = 0$ then
2:     return $1, 0, n$            // $1 \cdot n + 0 \cdot m = n = \gcd(n, m)$
3: else
4:     $x, y, r$ := extended-Euclid($m \bmod n, n$)
5:     return $y - \lfloor \frac{m}{n} \rfloor \cdot x, x, r$
```

- `long `***`find_inverse`***`(long a, long m)`:

Finds the modular multiplicative inverse of a modulo m based on the following algorithm:



```
inverse($a, n$):
Input: $a \in \mathbb{Z}_n$ and $n \geq 2$
Output: $a^{-1}$ in $\mathbb{Z}_n$, if it exists
 1: $x, y, d$ := extended-Euclid($a, n$)
 2: if $d = 1$ then
 3:     return $x \bmod n$        // $xa + yn = 1$, so $xa \equiv_n 1$.
 4: else
 5:     return "no inverse for $a$ exists in $\mathbb{Z}_n$."
```

- `int random_prime()`:
  Generates a random prime number.

- `boolean isPrime(int p)`:
  Check if a number is prime or not.

- `long[] generate_keys()`:
  Generate Keys method.

- `long modular_exponentiation(long b, long n, long m):`
  Find (b^n mod m) when we are dealing with big numbers, Same as algorithm 5 in 4.2 in the book
-
- `long string_to_int(String text):`
  Takes a string and converts each character to int , ex: Input:KBL, Output: 100111.

- `String int_to_String(long inttext):`
  Convert from int_to_String , ex: Input: 100111, Output: KBL

- `long[] encrypt(String plaintext, long e, long n):`
  Encryption method.

- `String decrypt(long[] ciphertext, long d, long n)`

<u>Notes:</u>

- Your code will include Class **main** for testing and Class **RSA. RSA** class is the important class that contains the previous methods.
- Since we will be dealing with big numbers, use long instead of int.
- Again, since the numbers are big, use modular_exponentiation instead of Math.pow.
- Do not use Class BigInteger.
- You must implement all previous methods.
- You may add as many auxiliary methods as needed.
- You should **print** the results **<u>of each step</u>** on RSA.
- You should write **exactly** the same name of each method.