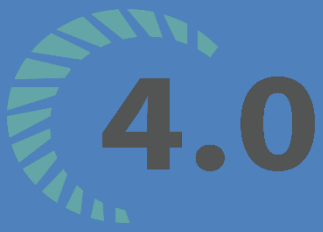
BỘ MÔN CÔNG NGHỆ TRI THỨC – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC QUỐC GIA TP HCM

NHẬP MÔN MÃ HÓA – MẬT MÃ



Sinh viên thực hiện: Nguyễn Đình Trí

GV phụ trách: Nguyễn Đình Thúc

ĐỒ ÁN CÁ NHÂN – THƯ VIỆN TÍNH SỐ NGUYÊN LỚN

HỌC KỲ I – NĂM HỌC 2020-2021

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

THÔNG TIN CHI TIẾT

Họ tên: Nguyễn Đình Trí

MSSV: 18120611

Lớp: Nhập môn mã hóa – mật mã 18_22

Tên đồ án: Thư viện tính số nguyên lớn

Loại: Đồ án cá nhân

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

YÊU CẦU ĐỒ ÁN

Loại bài tập	**☑ Lý thuyết ☐ Thực hành ☑ Đồ án ☐ Bài tập**
Ngày bắt đầu	**16/10/2020**
Ngày kết thúc	**16/11/2020**

A. Yêu cầu của đồ án

Viết 3 hàm:

- **GCD (bigInt a, bigInt b)**: trả về ước chung lớn nhất của 2 số nguyên lớn a và b.
- **mulMod (bigInt a, bigInt b, bigInt n)**: trả về kết quả của phép tính a * b trên vành n (a * b (mod n)) với a, b, n là các số nguyên lớn và 0 <= a, b < n.
- **powerMod (bigInt x, bigInt p, bigInt n)**: trả về kết quả của phép tính x ^ p trên vành n (x ^ p (mod n)) với x, p, n là các số nguyên lớn và x < n, p tùy ý.

Lưu ý: Tự thiết kế thư viện số nguyên lớn và các hàm liên quan để xử lý yêu cầu trên.

B. Thực hiện

1. Ngôn ngữ sử dụng: C#; ứng dụng chạy trên console.
2. Thiết kế lớp **bigInt** cho số nguyên lớn
    a. Biến *_bits* chứa dạng chuỗi nhị phân.
    b. Biến *_length* chứa độ dài của chuỗi nhị phân đó.
    c. Hàm tạo mặc định tạo chuỗi có độ dài là 1 ký tự 0; hàm tạo với tham số **String str** sẽ tạo ra một chuỗi nhị phân chứa trong trong **str** vào biến *_bits* và *_length* chính bằng độ dài của chuỗi **str** này; hàm tạo với tham số **int length** tạo ra chuỗi các số 0 với độ dài **length**.
    d. Các operator như +, -, %, <, >, ==, !=, <<, >> để hỗ trợ cho việc tính toán được cài đặt theo những phương pháp xử lý số nhị phân thông thường.
    e. Hàm **bool equalToZero()** trả về **true** nếu số nguyên lớn đó có giá trị 0 và **false** nếu không phải.
    f. Hàm **static bigInt standardize (bigInt a)** sẽ trả về số nguyên lớn được chuẩn hóa từ chuỗi nhị phân của số nguyên lớn **a** bằng cách xóa hết tất cả những kí tự **0** trước kí tự **1** đầu tiên của *a._bits* và cập nhật lại độ dài.

3. Lớp **MyRandom** được thiết kế theo mẫu **Singleton** giúp cho việc tạo số ngẫu nhiên giảm khả năng bị trùng lặp được sử dụng để tạo chuỗi bit trong hàm **bigIntGen (int bitLength)** nhằm tạo ngẫu nhiên 1 số nguyên lớn.
4. Những hàm quan trọng:
   a. **GCD (bigInt a, bigInt b)**
   Được thiết kế theo giải thuật sau:

```
g = 1;
while (a, b đều chẵn)
{
        a /= 2; // a >> 1
        b /= 2; // b >> 1
        g *= 2; // g << 1
} // a hoặc b lẻ
while (a > 0)
{
        while (a chẵn) a /= 2 // a lẻ
        while (b chẵn) b /= 2 // b lẻ
        t = |a – b| / 2;
        if (a >= b) a = t;
        else b = t;
}
g = g * b;
return g;
```

   b. **addMod (bigInt a, bigInt b, bigInt n)**
   Input: a, b $\in Z_n$, *n* $\in N^*$

```
if (a + b < n) return a + b;
else return a + b - n;
```

   c. **subMod (bigInt a, bigInt b, bigInt n)**
   Input: a, b $\in Z_n$, *n* $\in N^*$

```
if (a < b) return a - b + n;
return a - b;
```

   d. **mulMod (bigInt a, bigInt b, bigInt n)**
   Input: a, b $\in Z_n$, *n* $\in N^*$

```
if (b._bits [b._length - 1] = 1) result = x;
else result = 0;
for i = b._length - 2 to 1 do
{
        a = addMod (a, a, n);
```

```
        if (b._bits[i] = 1) result = addMod (result, a, n);
    }
    return result;
```

e. **powerMod (bigInt x, bigInt p, bigInt n)**

```
Input: a, b ∈ Zn, n, p ∈ N*
result = 1;
for i = 0 to n do
{
        result = mulMod(result, result, n);
        if (p[i] = 1) result = mulMod (result, x, n);
}
return result;
```

5. Cách nhập dữ liệu:
   a. Nhập dữ liệu từ file data.txt (gồm 4 dòng, mỗi dòng trong file là một chuỗi nhị phân lần lượt của x, y, p, n).
   b. Nhập dữ liệu bằng cách tạo ngẫu nhiên các chuỗi nhị phân của lần lượt x, y, p, n với độ dài được nhập bởi người dùng.

*Lưu ý: Dữ liệu phải thỏa điều kiện: giá trị **0 <= x, y < n**; **p tùy ý**. Nếu không thỏa điều kiện trên sẽ gây ra việc mất thời gian khi tính toán.*