T.P. (2da parte): Sistema criptográfico RSA

El *sistema criptográfico RSA* es un método para enviar un mensaje de un emisor a un receptor, codificado de tal manera que si el mensaje es interceptado por terceros, no puedan descifrar su contenido. El nombre proviene de las iniciales de sus creadores: Rivest, Shamir y Adleman.

Este sistema está basado en la siguiente propiedad matemática:

Lema: Sean p, q dos primos positivos distintos y sean $n = pq$ y $m = (p-1)(q-1)$. Finalmente, sean $d, e \in \mathbb{N}$ tales que

$$de \equiv 1 \pmod{m}.$$

Entonces, para todo $a \in \mathbb{Z}$,

$$a^{de} \equiv a \pmod{n}.$$

La demostración de este lema puede encontrarse por ejemplo en las notas *Álgebra I* de T. Krick.

¿Cómo funciona el sistema criptográfico RSA?

Alice quiere enviarle un mensaje a Bob (Alice y Bob son los nombres originales utilizados en la presentación del sistema RSA en 1978). Supongamos que el mensaje está dado por una lista de caracteres y a cada caracter le corresponde un entero entre 0 y 127 (ver las **Aclaraciones importantes** al final del enunciado).

Para comenzar, Bob debe generar por única vez, una *clave pública* y una *clave privada*, para lo que procede de la siguiente manera:

- Elige dos números primos distintos p y q y calcula $n = pq$ y $m = (p-1)(q-1)$. Si $n \leq 127$ debe empezar de vuelta con primos más grandes.
- Elige cualquier número e con $2 \leq e \leq m-2$ que sea coprimo con m y calcula $d \in \mathbb{N}$ tal que

  $$de \equiv 1 \pmod{m}.$$

  El número d se llama el *exponente de cifrado* y el número e se llama el *exponente de descifrado*.
- Publica el par (n, d) que es su *clave pública* y guarda para sí el par (n, e) que es su *clave privada* (no es necesario que Bob recuerde ni p ni q ni m).

De esta manera, Alice, al igual que todo el mundo, conoce la clave pública (n, d) de Bob pero desconoce su clave privada (n, e) (en realidad lo que desconoce es solamente el exponente de descifrado e). Para encriptar un mensaje, Alice procede de la siguiente manera:

- Arma la lista de números enteros que se forma reemplazando cada caracter por el número entre 0 y 127 que le corresponde.
- Reemplaza cada número a de su lista por el resto de a^d en la división por n.

La lista de números obtenida es el mensaje encriptado de Alice. Alice envía su mensaje encriptado a Bob, que para desencriptarlo procede de la siguiente manera:

- Reemplaza cada número b de la lista recibida por el resto de b^e en la división por n. De esta manera, recupera el número a original, ya que
$$b^e \equiv (a^d)^e \equiv a^{de} \equiv a \pmod{n}.$$
- Arma la lista de caracteres que se forma reemplazando cada número que acaba de obtener por el caracter que le corresponde.

Eventualmente, terceras personas pueden interceptar el mensaje encriptado que Alice le envía a Bob, pero no pueden desencriptarlo porque desconocen el exponente de descifrado e.

Por otro lado, cualquier persona que pueda factorizar n, obtendrá el valor de p y q, con lo cual podrá calcular el valor de m y e (sabiendo que $de \equiv 1 \pmod{m}$) y luego desencriptar los mensajes que Bob reciba; a esto se le dice *romper el código*. En efecto, la seguridad del sistema criptográfico RSA se basa en que hoy en día no se conocen algoritmos eficientes para factorizar números grandes.

Esta segunda parte del T.P. consiste en programar las siguientes funciones:

• `generarClaves :: Integer -> Integer -> ((Integer, Integer), (Integer, Integer))` que dados dos números primos p y q, genera un par que contiene una clave pública y una clave privada en el formato $((n, d), (n, e))$.

Aclaración: El exponente de descifrado e de la clave privada, coprimo con m, puede ser elegido **de cualquier manera** que consideren conveniente; por ejemplo: lo más chico posible, el primero posible a partir de 33, lo más grande posible, etc (recordar la condición $2 \leq e \leq m - 2$).

Sugerencia: Para calcular el exponente de cifrado d de la clave pública, pueden utilizar las funciones de la clase 9 y/o de la clase 10. Notar que hallar d es equivalente a resolver la ecuación de congruencia
$$eX \equiv 1 \pmod{m}.$$

• `encriptar :: (Integer, Integer) -> String -> [Integer]` que dada una clave pública y un mensaje, lo reemplaza por la lista de enteros que lo encripta.

Recuerdo: `String` es lo mismo que `[Char]`.

• `desencriptar :: (Integer, Integer) -> [Integer] -> String` que dada una clave privada y una lista de enteros que encripta un mensaje, lo desencripta.

• Ejercicio opcional: Romper el código asociado a la clave pública $(100337, 60953)$, desencriptar la siguiente pregunta:

$$[33706, 38913, 58255, 99961, 77756, 23881, 220, 77756, 1606, 38913, 77756, 78982, 18800,$$
$$91658, 91658, 58255, 77756, 96593, 58255, 438, 22839, 28700, 18800, 1606, 58255, 48389]$$

encriptar la respuesta, y ponerla como comentario en el T.P.

Aclaraciones importantes:

• En Haskell hay funciones para obtener el número que le corresponde a cada caracter y viceversa. Para poder acceder a ellas, hay que incluir la línea

```
import Data.Char (ord, chr)
```

al comienzo del archivo .hs. Luego se pueden utilizar las funciones

```
ord ::  Char -> Int
```

y

```
chr ::  Int -> Char
```

que proporcionan una correspondencia entre caracteres y números enteros entre 0 y 127.

Notar que estas funciones utilizan el tipo `Int`, luego es necesario combinarlas con las funciones `fromIntegral` para convertir `Int` en `Integer` y `fromInteger` para convertir `Integer` en `Int`.

• El T.P. es individual.

• Cada alumno tiene que entregar un único archivo .hs, cuyo nombre esté formado por su apellido, su numero de libreta (sin la barra) y su numero de grupo actual (luego de la reorganización de los grupos), separados por guión bajo (sin espacios); por ejemplo: Fernandez_08420_grupo4.hs. La entrega se hace subiendo el archivo al campus.

• No se pueden modificar los nombres y las signaturas de las funciones que se pide programar.

• Pueden programar todas las funciones auxiliares que consideren necesarias, pero el código de estas funciones debe ser parte de la entrega, incluso si utilizan funciones como por ejemplo `emcd` que es parte de la lista de ejercicios de la clase 9. Las funciones auxiliares deben tener nombres declarativos, y de ser necesario, deben incluirse también comentarios de aproximadamente dos o tres líneas para explicar qué hacen.

• La **fecha y hora límite** para la entrega es el **Domingo 26 de julio a las 23:59**.

• Si tienen dudas con respecto al enunciado de esta segunda parte del T.P. (es decir, si no les quedan claras las consignas), pueden hacer consultas enviando un mail a la lista de docentes algebra1-doc (arroba) dc.uba.ar. **No hacer consultas en los foros ni mandando mails a la lista de alumnos.**

• El T.P. debe resolverse utilizando los temas que se dictaron en el Taller de Álgebra este cuatrimestre.