GIIT_AMA_2223_Practica5_alokenveo_jelanang

Práctica 5. Teoría de Números: Congruencias

Temporalización de la práctica: Lunes 21 de noviembre y Lunes 28 de noviembre de 2022

Entrega de la práctica: Desde el 28 de noviembre hasta el 5 de diciembre (ver tarea en el campus virtual o agenda de la asignatura)

Instrucciones:

1. Haz una copia de la hoja pública y renómbrala: si tu correo es **mariomp@alumnos.unex.es** añade al final del título **_mariomp**, por ejemplo GIIT_AMA_2223_Practica5_mariomp

- Para cambiar el nombre pulsa en el título de la hoja (arriba del todo, entre el logo de Sage y el menú "Archivo/File...")

2. Comparte la hoja de trabajo con el usuario **mariomp2223** mediante el botón Compartir/Share de arriba a la derecha. Si lo hacéis en pareja, compartid la hoja con el usuario de tu compañero (ambos usuarios separados por comas), así como añadir también el usuario en el título de la hoja (separados por _).

3. Completa la primera celda y trabaja la práctica.

4. Cuando hayas terminado, haz una copia en un único fichero PDF y ponlo en el campus virtual (Si lo hacéis en pareja, basta que lo suba uno). **Esa será la versión que se evaluará**. La hoja no se considera entregada si no se ha renombrado y compartido (pasos 1 y 2).

- Para generar el PDF lo más sencillo es usar el botón Imprimir/Print de arriba e imprimir la nueva página a fichero.

5. Una vez subido el PDF al campus virtual, no podrá modificarse esta hoja de trabajo. **Hacerlo conllevará la calificación de 0 en esta práctica**.

6. Los ejercicios a entregar se representan en Rojo y deben estar correctamente explicados. Los ejercicios indicados con ** **NO** serán obligatorios, pero aquellos que los hagan podrán ir sumando por cada uno de ellos 0.1 puntos adicionales en la calificación de la práctica.

Alumno/s: Jose Luis Obiang Ela Nanguan, Alfredo Mituy Okenve

En primer lugar, recordemos que fijado $m \in \mathbb{N}$, se dice que $a, b \in \mathbb{Z}$ son ***congruentes módulo*** m

$$a \equiv b \bmod m$$

si m es divisor de $a-b$, o equivalentemente, si a y b tienen el mismo resto al dividir por m.

Por ejemplo, veamos que $5 \equiv 1 \mod 4$, pues 5 y 1 tienen el mismo resto al dividir por 4.

```
print "Resto de dividir 5 por 4:", 5%4
print "Resto de dividir 1 por 4:", 1%4
```

```
Resto de dividir 5 por 4: 1
Resto de dividir 1 por 4: 1
```

Pero del mismo modo se puede decir que $5 \equiv -3 \mod 4$

```
print "Resto de dividir 5 por 4:", 5%4
print "Resto de dividir -3 por 4:", (-3)%4
```

```
Resto de dividir 5 por 4: 1
Resto de dividir -3 por 4: 1
```

Por tanto, hacer congruencias módulo m es crear un conjunto de números enteros que tienen todos el mismo resto al dividir por m, a este conjunto de números se les denota como $\mathbb{Z}/m\mathbb{Z}$, donde cada número $n \in \mathbb{N}$ será congruente con un conjunto de números enteros, a este conjunto se le llama ***Clase de equivalencia*** de n, denotada como $[n]_m$. De modo que el conjunto

$$\mathbb{Z}/m\mathbb{Z} = \{[n]_m : n = 0, 1, 2, \ldots, m-1\}$$

Por ejemplo,

$$\mathbb{Z}/4\mathbb{Z} = \{[0]_4, [1]_4, [2]_4, [3]_4\}$$

Puede observarse que los valores de n coinciden con los posibles valores que puede tomar el resto al dividir un número entero por 4 (Algoritmo de la División). De modo que como 5 tenía como resto 1, entonces $5 \equiv 1 \mod 4$, esto es, $5 \in [1]_4$.

Sage permite trabajar de manera cómoda con congruencias, pues permite operar con clases de equivalencia sobre un conjunto $\mathbb{Z}/m\mathbb{Z}$, a partir de la función ***IntegerModRing(m)*** que permite crear dicho conjunto.

```
m = 4
Zm = IntegerModRing(m)
show(Zm)
```

```
0123
```

Observamos que muestra los representantes (n) de cada clase de equivalencia $[n]_m$. A partir de Zm podemos operar con clases de equivalencia, pues ***Zm(n)*** será la clase de equivalencia $[n]_m$.

```
print "[",Zm(2),"] + [",Zm(3),"] = [", Zm(2)+Zm(3), "]"
print "[",Zm(1),"] + [",Zm(-1),"] = [", Zm(1)+Zm(-1), "]"
print "[",Zm(2),"] * [",Zm(2),"] = [", Zm(2)*Zm(2), "]"
```

```
[ 2 ] + [ 3 ] = [ 1 ]
[ 3 ] + [ 3 ] = [ 2 ]
[ 2 ] * [ 2 ] = [ 0 ]
```

Si nos fijamos en la segunda línea, hemos escrito Zm(-1), es decir, la clase de equivalencia de -1 módulo 4, sin embargo Sage muestra la clase de equivalencia de 3 módulo 4. Esto se debe a que Sage muestra como representante al valor positivo del resto (0,1,2,3), ya que el resto de dividir -1 por 4 es 3.

```
(-1)%4
```

```
3
```

Además, podemos resolver ecuaciones en congruencias muy sencillas, por ejemplo veamos si existe una clase $[x]$ que cumpla que $[3][x] = [1]$ en $\mathbb{Z}/4\mathbb{Z}$. Si nos damos cuenta, estamos buscando un entero x tal que $3x \equiv 1 \bmod 4$. Uno puede pensar que ese número es $1/3$, pero debe ser entero, y éste es racional. Realmente, estamos buscando el inverso de 3 en $\mathbb{Z}/4\mathbb{Z}$.

Para buscar el inverso de un número entero a en $\mathbb{Z}/m\mathbb{Z}$, es necesario comprobar si existe, y lo hará si $mcd(a,m) = 1$, esto es, una clase de equivalencia $[a]_m$ tendrá inverso si $mcd(a,m) = 1$, y en tal caso será aquella clase $[b]$ tal que $[a][b] = [1]$.

Utilizando la función anterior, es fácil calcular dicho inverso (en caso de que exista) simplemente indicando

$$Zm(1)/Zm(a)$$

```
a,m = 3,4
Zm = IntegerModRing(m)
Zm(1)/Zm(a)
```

```
3
```

Podemos ver que el inverso de la clase [3] es ella misma, pues [3][3] = [1] en módulo 4.

```
Zm(3)*Zm(3)
```

```
1
```

Ejercicio 1. Determina el número entero x (en caso de que exista) en cada congruencia.

a) $89567 \equiv x \bmod 7$

b) $90783 \equiv x \bmod 24$

c) $29x \equiv 1 \bmod 3$

d) $345678 \equiv x \bmod 13$

e) $18x \equiv 1 \bmod 5$

f) $12x \equiv 2 \bmod 4$

g) $8932145678x \equiv 1 \bmod 2$

```
#Realiza el ejercicio en esta celda
#Función reutilizable en toda la práctica para hallar el resto de una congruencia pasados a y m como parámetros
#a)
def resto(a,m):
    x = a%m; print "x =",x #Hallamos el resto

a,m=89567,7
resto(a,m)
```

```
    x = 2
```

```
#b)
a,m = 90783,24
resto(a,m)    #Ejecutamos la función pasando la base y el módulo para hallar b o el resto
```

```
    x = 15
```

```
#c)
#función reutilizable para hallar el inverso de una congruencia.
def inverso(a,m,b):
```

```
    x=0
    if gcd(a, m) == 1:    #Si existe inverso
        Zm=IntegerModRing(m) #Obtenemos la lista de congruencias del modulo
        x=Zm(b)/Zm(a) #Obtenemos el inverso
    else:    #Si no existe inverso, no existe solución
        print "No existe solución\n",m,"y",a,"NO son primos entre si, por lo tanto, no existe el inverso de",a,"
en modulo",m
    return x

a,m,b = 29,3,1
inverso(a,m,b)
```

```
    2
```

```
#d)
a,m=345678,13
resto(a,m)
```

```
    x = 8
```

```
#e)
a,m,b=18,5,1
inverso(a,m,b)
```

```
    2
```

```
#f)
a,m,b=12,4,2
inverso(a,m,b)
```

```
    No existe solución
    4 y 12 NO son primos entre si, por lo tanto, no existe el inverso de
    12  en modulo 4
    0
```

```
#g)
a,m,b=8932145678,2,1
inverso(a,m,b)
```

```
    No existe solución
    2 y 8932145678 NO son primos entre si, por lo tanto, no existe el
    inverso de 8932145678  en modulo 2
    0
```

Con esta herramienta de Sage, también se pueden determinar de modo sencillo las congruencias de las potencias de números respecto a un módulo. Por ejemplo, veamos cuánto es 12^{26} en módulo 23.

```
a = 12^26
print a
m = 22
Zm = IntegerModRing(m)
Zm(a)
```

```
114475459972882815552155811184
12
```

Podemos comprobar fácilmente que trabajar con congruencias, podemos determinar 12^{26} módulo 23 directamente, como acabamos de hacer, o bien, determinar 12 módulo 23, y a continuación realizar la potencia.

```
b = Zm(12); print "[12] es [",b,"] mod 23"
Zm(b)^26
```

```
[12] es [ 12 ] mod 23
12
```

Ejercicio 2. Sea DNI tu número de DNI (comenzando en un número no nulo), determina $DNI \equiv x \mod 23$.

```
#Realiza el ejercicio en esta celda
a,m=7369285,23
print "Mi DNI es",a
resto(a,m)    #Obtenemos el resto
```

```
Mi DNI es 7369285
x = 16
```

Ejercicio 3. Sea DNI tu número de DNI (comenzando en un número no nulo), determina $DNI^{123456789}$ en $\mathbb{Z}/23\mathbb{Z}$.

```
#Realiza el ejercicio en esta celda
a,m=7369285,23
Zm=IntegerModRing(m) #Obtenemos la lista de congruencias de m=23
b=Zm(a)
print Zm(b)^123456789  #Realizamos la potencia
```

```
6
```

Ejercicio 4. Determina 61 elevado a 123456789 en $\mathbb{Z}/8\mathbb{Z}$. ¿Cuál es el valor de k tal que $61^k \equiv 1 \bmod 8$?

```
#Realiza el ejercicio en esta celda
a=61
m=8
Zm=IntegerModRing(m)
b=Zm(a)
print "El valor a determinar es igual a",Zm(b)^123456789

#Para hallar k creamos una funcion que busque el valor deseado,
def getExponente(a,m):
    k=1
    while (a^k)%m!=1:
        k=k+1
    print "k es igual a",k

getExponente(a,m)
```

```
El valor a determinar es igual a 5
k es igual a 2
```

Ejercicio 5. Determina 893 elevado a 32456781 en $\mathbb{Z}/11\mathbb{Z}$. ¿Qué valor toma k para que $893^k \equiv 1 \bmod 11$?

```
#Realiza el ejercicio en esta celda
a=893
m=11
Zm=IntegerModRing(m)
b=Zm(a)
print "El valor a determinar es igual a",Zm(b)^32456781

getExponente(a,m)
```

```
El valor a determinar es igual a 2
k es igual a 10
```

Ejercicio 6. Crea una función que reciba dos números a,m y muestre por pantalla las potencias a^k y su congruencia en módulo m, hasta que encuentre aquel valor de k tal que $a^k \equiv 1 \mod m$.

```
#Realiza el ejercicio en esta celda
#Haré una funcion que vaya recorriendo números uno por uno hasta dar con el que cumple las condiciones que se
piden
def listPoten(a,m):
    k=1  #Este es el valor mínimo que puede tener el nº k
    while (a^k)%m!=1:       #Pongo un while para que no pare de ejecutar lo que está dentro hasta que el valor
que hace que se cumpla la condición
        print "a^",k,"=",a^k,";Su congruencia en m es",(a^k)%m        #Y de paso voy enseñando cada iteración
que se hace
        k=k+1
    print "\nk final es igual a",k
```

```
#AÑADO PRUEBA DE QUE FUNCIONA
a=893
m=11
listPoten(a,m)
```

```
a^ 1 = 893 ;Su congruencia en m es 2
a^ 2 = 797449 ;Su congruencia en m es 4
a^ 3 = 712121957 ;Su congruencia en m es 8
a^ 4 = 635924907601 ;Su congruencia en m es 5
a^ 5 = 567880942487693 ;Su congruencia en m es 10
a^ 6 = 507117681641509849 ;Su congruencia en m es 9
a^ 7 = 452856089705868295157 ;Su congruencia en m es 7
a^ 8 = 404400488107340387575201 ;Su congruencia en m es 3
a^ 9 = 361129635879854966104654493 ;Su congruencia en m es 6

k final es igual a 10
```

Ecuaciones con Congruencias

Una ecuación con congruencias es de la forma

$$ax \equiv b \bmod m$$

que tiene solución si y solo si $d = mcd(a, m)$ es divisor de b. En el caso de existir, la solución es:

$$x_0 = (a/d)^{-1}(b/d)$$

donde $(a/d)^{-1}$ será el inverso de a/d en módulo m/d. Además, esta ecuación tiene exactamente d soluciones:

$$x \equiv x_0 + (m/d)k, k = 0, 1, \ldots, d-1, \bmod m$$

En particular, si $mcd(a, m) = 1$, entonces para todo entero b la ecuación tiene una única solución:

$$x \equiv a^{-1}b \bmod m$$

Veamos un ejemplo: resolveremos la ecuación con congruencias $2x \equiv 6 \bmod 10$. Lo primero que debemos comprobar es si tiene solución.

```
gcd(2,10)
```

```
2
```

Vemos que el mcd(2,10) es 2, y que es divisor de 6, por lo que efectivamente la ecuación con congruencias tiene solución. Para ello, debemos calcular a/d, b/d y m/d. De forma que la ecuación

$$2x \equiv 6 \bmod 10$$

es equivalente a

$$(a/d)x \equiv (b/d) \bmod m/d$$

, con solución

$$x_0 = (a/d)^{-1}(b/d)$$

.

```
a,b,m = 2,6,10
d = gcd(a,m)
a/d,b/d, m/d
```

```
(1, 3, 5)
```

Esto es, la ecuación con congruencias sería

$$1x \equiv 3 \bmod 5$$

por lo tanto

$$x \equiv 3 \bmod 5$$

De modo que,

$$x_0 = 3 \bmod 5$$

Pero nuestro objetivo es resolver la ecuación con congruencias $2x \equiv 6 \bmod 10$, luego se puede trasladar esta solución a módulo 10. En particular, esta ecuación tiene $d = 2$ soluciones:

$$x \equiv x_0 + (m/d)k, k = 0, 1, \bmod m$$

```
x0 = 3
[x0 + (m/d)*k for k in [0..(d-1)]]
```

```
[3, 8]
```

Ejercicio 7. Resuelve la ecuación con congruencias $3x \equiv 1 \bmod 19$.

```
#Realiza el ejercicio en esta celda
a,m,b=3,19,1
inverso(a,m,b)
```

```
13
```

Ejercicio 8. Resuelve la ecuación con congruencias $28x \equiv 12 \bmod 8$.

```
#Realiza el ejercicio en esta celda
a,m,b=28,8,12
inverso(a,m,b)
```

```
8 y 28 NO son primos entre si, por lo tanto, no existe el inverso de
28  en modulo 8
0
```

Ejercicio 9. Teorema Chino de los Restos: Sean $m_1, m_2, \ldots, m_n$ primos entre sí. Entonces el sistema de ecuaciones con congruencias:

$$\left.\begin{aligned} x &\equiv b_1 \bmod m_1 \\ x &\equiv b_2 \bmod m_2 \\ &\cdots \\ x &\equiv b_n \bmod m_n \end{aligned}\right\}$$

tiene solución única módulo $m = m_1 \cdot m_2 \cdots m_n$

$$x \equiv b_1 M_1 y_1 + b_2 M_2 y_2 + \ldots + b_n M_n y_n \bmod m$$

donde $M_k = m/m_k$ e y_k es el inverso de M_k en módulo m_k.

Crea una función que resuelva un sistema de ecuaciones con congruencias utilizando el Teorema Chino de los Restos. Para ello, la función debe recibir una lista con los módulos, m_k, y una lista con los enteros b_k (k = 1,2,...,n), y debe devolver la solución del sistema (x) y el módulo (m).

```
#Realiza el ejercicio en esta celda
def solveSistEqua(lista_modulos,lista_enteros):
    m = 1
    size = len(lista_modulos)-1    #Obtenemos el tamaño de la lista
    for i in [0..size]:    #Recorremos la lista
        m = m * lista_modulos[i]   #Vamos acumulando el resultado de la multiplicación de cada uno de los
módulos
    x=0
    for i in [0..size]:    #Recorremos la lista de modulos y la lista de enteros
        M = int(m/lista_modulos[i])    #Obtenemos el modulo M1, M2, M3, ...
        y = int(inverso(M,lista_modulos[i],1)) #Obtenemos y1, y2, y3, ...
        b = int(lista_enteros[i]) #Obtenemos el entero b1, b2, b3, ...
```

```
        print "b",i,":",b,", M",i,":",M," y",i,":",y
        x=x+b*M*y  #Vamos acumulando el resultado de la multiplicación de b,M e y
    return "El valor de x es:",x," y el modulo producto es:",m

#Ejemplo:

modulos=[4,7,5] #lista modulos
enteros=[4,2,11]    #Lista de valores de b
solveSistEqua(modulos,enteros)
```

```
b 0 : 4 , M 0 : 35  y 0 : 3
b 1 : 2 , M 1 : 20  y 1 : 6
b 2 : 11 , M 2 : 28  y 2 : 2
('El valor de x es:', 1276, ' y el modulo producto es:', 140)
```

**** Ejercicio 10. Modifica la función anterior, para que resuelva el sistema de ecuaciones con congruencias:**

$$\left.\begin{aligned} a_1 x &\equiv b_1 \bmod m_1 \\ a_2 x &\equiv b_2 \bmod m_2 \\ &\dots \\ a_n x &\equiv b_n \bmod m_n \end{aligned}\right\}$$

donde $mcd(a_k, m_k) = 1$ y la solución única en módulo $m = m_1 m_2 \cdots m_k$ es

$$x \equiv a_1^{-1} b_1 M_1 y_1 + a_2^{-1} b_2 M_2 y_2 + \ldots + a_n^{-1} b_n M_n y_n \bmod m$$

donde a_k^{-1} es el inverso de a_k en módulo m_k.

```
#Realiza el ejercicio en esta celda
def solveSistEqua(lista_modulos,lista_enterosA, lista_enterosB):
    m = 1
    size = len(lista_modulos)-1    #Obtenemos el tamaño de la lista
    for i in [0..size]:
        m = m * lista_modulos[i]     #Vamos acumulando el resultado de la multiplicación de cada uno de los
módulos
```

```
    print "Modulo producto de modulos: ",m
    x=0
    for i in [0..size]:
        M = int(m/lista_modulos[i])    #Obtenemos el modulo M1, M2, M3, ...
        y = int(inverso(M,lista_modulos[i],1)) #Obtenemos y1, y2, y3, ...
        b = int(lista_enterosB[i]) #Obtenemos el entero b1, b2, b3, ...
        a = int(inverso(lista_enterosA[i],lista_modulos[i],lista_enterosB[i])) #Obtenemos el inverso del entero
a1, a2, a3, ...
        print "Inverso de a",i,":",a,", b",i,":",b,", M",i,":",M," y",i,":",y
        x=x+a*M*y   #Vamos acumulando el resultado de la multiplicación del inverso de a,M e y
    print "El valor de x es:",x
    return  "El menor numero natural positivo solucion del sistema de ecuaciones en congruencias es:",x%m

#Ejemplo:

modulos=[4,7,5] #lista modulos
enterosB=[4,2,11] #Lista de valores de b
enterosA=[9,4,12]    #Lista de valores de a
solveSistEqua(modulos,enterosA, enterosB)
```

```
Modulo producto de modulos:  140
Inverso de a 0 : 0 , b 0 : 4 , M 0 : 35  y 0 : 3
Inverso de a 1 : 4 , b 1 : 2 , M 1 : 20  y 1 : 6
Inverso de a 2 : 3 , b 2 : 11 , M 2 : 28  y 2 : 2
El valor de x es: 648
('El menor numero natural positivo solucion del sistema de
ecuaciones en congruencias es:',
 88)
```

Ejercicio 11. Resuelve el sistema de ecuaciones con congruencias:

$$\left.\begin{aligned} 3x+9 &\equiv 2 \bmod 5 \\ 2x-5 &\equiv 1 \bmod 3 \end{aligned}\right\}$$

(La solución del sistema se encuentra en el enunciado de los ejercicios 10 y 11).

```
#Realiza el ejercicio en esta celda
modulos=[5,3] #lista modulos
enterosB=[2-9,1+5] #Lista de valores de b
enterosA=[3,2]    #Lista de valores de a
solveSistEqua(modulos,enterosA, enterosB)
```

```
Modulo producto de modulos:  15
Inverso de a 0 : 1 , b 0 : -7 , M 0 : 3  y 0 : 2
Inverso de a 1 : 0 , b 1 : 6 , M 1 : 5  y 1 : 2
El valor de x es: 6
('El menor numero natural positivo solucion del sistema de
ecuaciones en congruencias es:',
 6)
```

Teorema de Euler

Otra estrategia para determinar el resto de dividir potencias de números muy grandes por un número entero, es hacer uso del Teorema de Euler. Para ello, es necesario utilizar el ***Indicador de Euler*** de un número natural no nulo, m, $\phi(m)$, que recordemos que es el número de enteros positivos entre 1 y m que son primos con m. Para ello, Sage tiene implementado la función ***euler_phi(m)***.

Por ejemplo, veamos cuál es el indicador de Euler de varios números.

```
euler_phi(8),euler_phi(5),euler_phi(1)
```

```
(4, 4, 1)
```

Cuando p es un número primo, recordemos que $\phi(p) = p - 1$, o si $mcd(a, b) = 1$, entonces $\phi(ab) = \phi(a)\phi(b)$

```
p = 13; euler_phi(8319)
```

```
5336
```

```
a,b = 12,17
euler_phi(12*17), euler_phi(12)*euler_phi(17)
```

```
(64, 64)
```

Además, si p es primo, entonces

$$\phi(p^r) = p^r \left(1 - \frac{1}{p}\right)$$

Por ejemplo, $8 = 2^3$, entonces

$$\phi(8) = 2^3 \left(1 - \frac{1}{2}\right)$$

```
2^3*(1-1/2)
```

```
4
```

Teorema de Euler: Sean a, m enteros con $m \geq 1$. Si $mcd(a, m) = 1$, entonces

$$a^{\phi(m)} \equiv 1 \bmod m$$

Congruencia de Fermat: Sea p un número primo. Si un número entero a no es múltiplo de p, entonces

$$a^{p-1} \equiv 1 \bmod p$$

Apliquemos ambos resultados con algún ejemplo. Vamos a determinar el resto de dividir 3^{34291} por 5, es decir, nos interesa conocer a qué número es congruente 3^{34291} en módulo 5.

```
a,m = 3,5
pot = 34291
```

Podemos observar que $m = 5$ es un número primo, y que $a = 3$ no es múltiplo de 5, por lo que se puede aplicar la Congruencia de Fermat, de modo que

```
a^(m-1)
```

```
81
```

81 es congruente con 1 en módulo 5 (veámoslo).

```
81%5
```

```
1
```

Por lo tanto,

$$3^4 = 81 \equiv 1 \bmod 5$$

luego por las propiedades de las potencias podríamos escribir el exponente como

$$34291 = 4k + r$$

de tal modo que

$$3^{34291} = 3^{4k} \cdot 3^r$$

y como

$$3^{4k} = (3^4)^k \equiv 1^k = 1 \bmod 5$$

entonces

$$3^{34291} \equiv 3^r \bmod 5$$

por lo que necesitaríamos encontrar el valor de r. Pero éste es el resto de dividir 34291 por 4, es decir, es el entero al que es congruente 34291 en módulo 4.

```
r = 34291%4
print "r = ",r
```

```
r =  3
```

Por tanto,

$$3^{34291} \equiv 3^r = 3^3 \bmod 5$$

```
(3^3)%5
```

```
2
```

Es decir, $3^{34291} \equiv 2 \bmod 5$, luego el resto de dividir 3^{34291} por 5 es 2.

Ejercicio 12. Determina el resto de dividir 23^{84292} por 7. Aplica el Teorema de Euler o la Congruencia de Fermat, según sea el caso.

```
#Realiza el ejercicio en esta celda

#Vamos a hacer una función general que hay el resto para cada uno de los casos y luego aplicaremos según
convenga
def getResto(a,m,n,tipo):
    if tipo==0:
        if a%m!=0: #Introducimos esta comprobación para verificar que a no es múltiplo del módulo y así cumplir
las condiciones de la Congruencia
            print "En este caso aplicaremos la Congruencia de Fermat:"
            b=m-1 #Siendo m un nº primo el exponente de "b" será m-1
            r=n%b
            x=(a^r)%m
        else:
            print "No cumple las condiciones necesarias para aplicar la Congruencia de Fermat"
    else:
        if gcd(a,m)==1: #Introducimos esta comprobación para verificar que a y m cumplen las condiciones del
Teorema de Euler
            print "En este caso aplicaremos el Teorema de Euler:"
            b=euler_phi(m) #Aplicando el teorema de Euler lo que haremos es hallar el indicador de Euler
            r=n%b
            x=(a^r)%m
        else:
            print "No cumple las condiciones necesarias para aplicar el Teorema de Euler"
    print "El resto es igual a",x


#Dado que el módulo es un número primo aplicaremos la congruencia de Fermat
a,m,n=23,7,84292   #Introducimos los datos que nos proporciona el enunciado
tipo=0            #Ponemos tipo igual a 0 para que aplique la Congruencia de Fermat
getResto(a,m,n,tipo)
```

```
En este caso aplicaremos la Congruencia de Fermat:
El resto es igual a 2
```

Ejercicio 13. Determina el resto de dividir $123^{123456789}$ por 8. Aplica el Teorema de Euler o la Congruencia de Fermat, según sea el caso.

```
#Realiza el ejercicio en esta celda
#Hacemos uso de la fución creada en el ejercicio anterior
a,m,n=123,8,123456789   #Introducimos los datos que nos proporciona el enunciado
tipo=1                 #Ponemos tipo igual a 0 para que aplique el Teorema de Euler
getResto(a,m,n,tipo)
```

```
En este caso aplicaremos el Teorema de Euler:
El resto es igual a 3
```