PRÁCTICAS DE LA ASIGNATURA MATEMÁTICA DISCRETA.
Ingenierías Técnicas de Informática de Gestión y de Sistemas.
2º Curso. Primer Cuatrimestre.
Curso 2006-2007

Tema 2: TEORÍA DE NÚMEROS.

1. Elige uno de los apartados entre los ejercicios $3-b$), $4-a$), $4-c$), $5-b$) y $5-c$). Crea una función que reciba un número n y devuelva el valor de la expresión. La función ha de calcular la expresión por recursividad, basándose en la demostración por inducción. (Opcional)

2. Crea una función que reciba dos números a y b y devuelva su máximo común divisor. Utiliza para ello el algoritmo de Euclides. Para probar que el algoritmo de Euclides devuelve el máximo común divisor, se debería demostrar por inducción, basada en el número de pasos. Se valorará que la función esté programada de modo recursivo (basándose en que $mcd(a,b) = mcd(b,r)$, donde r es el resto de dividir a entre b). (Todos los grupos)

3. Crea una función que reciba dos números a y b y devuelva x e y tales que $mcd(a,b) = xa + yb$. De nuevo se valorará positivamente que la función sea recursiva. (Todos)

4. Crea una función que reciba dos números a y b y devuelva el mínimo común múltiplo calculado a partir del mcd. (Grupo 6,7,14,16)

5. Crea una función que reciba dos números a y b y devuelva $true$ si son primos entre sí y $false$ en caso contrario. Para ello llamará a la función que implementa el algoritmo de Euclides.

   (Grupo 6,7,14,16)

6. Crea una función que reciba un número n y devuelva los números primos menores o iguales a n. Utiliza para ello la criba de Eratóstenes. (Grupo 4,10,19)

7. Crea una función que reciba un número y devuelva $true$ si el número es primo y $false$ si no lo es. (Grupo 1,5,8)

8. Crea una función que reciba un número n y devuelva su descomposición en factores primos. Para devolver la decomposición en factores primos, se puede utilizar dos vectores (uno para los factores primos y otro para su multiplicidad) o una lista enlazada. (Opcional)

9. Crea una función que reciba dos números a y b, llame a la función del apartado 8 para calcular la descomposición en factores primos de cada uno y delvuelva el máximo común divisor de a y b. Puedes suponer que la función del apartado 8 ya está creada y responde a la siguiente declaración:

   ```
   void factores(int n, int* primos, int* coeficientes, int& k);
   ```

   donde $primos$ es un vector de k componentes conteniendo los números primos que dividen a n y $coeficientes$ es un vector de k componentes conteniendo la multiplicidad con que cada primo divide a n. (Grupo 2,11)

10. Crea una función que reciba dos números a y b, llame a la función del apartado 8 para calcular la descomposición en factores primos de cada uno y devuelva el mínimo común múltiplo de a y b. Puedes suponer que la función del apartado 8 ya está creada y responde a la siguiente declaración:

    ```
    void factores(int n, int* primos, int* coeficientes, int& k);
    ```

    donde *primos* es un vector de k componentes conteniendo los números primos que dividen a n y $coeficientes$ es un vector de k componentes conteniendo la multiplicidad con que cada primo divide a n. (Grupo 12,13,15)

11. Crea una función que reciba dos números a y b, llame a la función del apartado 8 para calcular la descomposición en factores primos de cada uno y *true* si a y b son primos entre sí y 0 en caso contrario (para ver si son primos, usará la descomposición en factores primos). Puedes suponer que la función del apartado 8 ya está creada y responde a la siguiente declaración:

    ```
    void factores(int n, int* primos, int* coeficientes, int& k);
    ```

    donde *primos* es un vector de k componentes conteniendo los números primos que dividen a n y $coeficientes$ es un vector de k componentes conteniendo la multiplicidad con que cada primo divide a n. (Grupo 3,9)

12. Crea una función que reciba los coeficientes a y b y el término independiente c de la ecuación diofántica lineal $ax + by = d$ y devuelva *true* en caso de que tenga solución y $false$ en caso de que no la tenga. Además, si tiene solución devolverá todas las soluciones (para ello utiliza cuatro variables: x, y, dx, dy y devuelve en ellas valores de modo que todas las soluciones sean $x + k\ dx$, $x + k\ dy$, para todo $k \in \mathbb{Z}$). (Grupo 9,12,13)

13. Crea una función que reciba un número n y devuelva las soluciones de la ecuación $x^2 - y^2 = n$. Puedes devolver las soluciones como dos vectores o como dos listas enlazadas. (Grupo 3,15)

14. Crea una función que reciba un número n y devuelva las ternas pitagóricas tales que $0 < x, y, z < n$. Hazlo usando la ecuación $x^2 + y^2 = z^2$. Puedes usar las propiedades de las ternas, pero no la expresión de las soluciones. (Grupo 10,11)

15. Crea una función que reciba un número n y devuelva las ternas primitivas pitagóricas tales que $0 < x, y, z < n$. Hazlo usando la expresión de las soluciones. Se valorará positivamente si la función devuelve todas las ternas pitagóricas en lugar de sólo las primitivas. (Grupo 2,19)

16. Crea una función que reciba dos números enteros a, b, m, con $m > 0$ y devuelva las soluciones (módulo m) de $ax \equiv b \mod m$. (Grupo 1,5)

17. Crea una función que reciba cuatro números enteros b_1, b_2, m_1, m_2, con m_1 y m_2 primos entre sí y devuelva las soluciones (módulo $m_1 m_2$) de

$$\begin{cases} x \equiv b_1 \mod m_1 \\ x \equiv b_2 \mod m_2 \end{cases}$$

(Grupo 4,8)

18. Crea una función que reciba seis números enteros a_1, a_2, b_1, b_2, m_1, m_2, con a_1 y m_1, a_2 y m_2, y m_1 y m_2 primos entre sí y devuelva las soluciones (módulo m_1m_2) de

$$\begin{cases} a_1x \equiv b_1 \mod m_1 \\ a_2x \equiv b_2 \mod m_2 \end{cases}$$

(Grupo 16)

19. Crea una función que reciba seis números enteros a_1, a_2, b_1, b_2, m_1, m_2, y devuelva las soluciones (módulo m_1m_2) de

$$\begin{cases} a_1x \equiv b_1 \mod m_1 \\ a_2x \equiv b_2 \mod m_2 \end{cases}$$

(Grupo 14)

20. Crea una función que reciba siete números enteros, a_1, a_2, b_1, b_2, m, (puedes agruparlos en dos o tres vectores más m o en una matriz más m) y devuelva las soluciones (módulo m) de

$$\begin{cases} a_1x \equiv b_1 \mod m \\ a_2x \equiv b_2 \mod m \end{cases}$$

(Grupo 6,7)

21. Crea una función que reciba un número n y devuelva $\phi(n)$ (usa para ello la definición de $\phi(n)$.

(Grupo 1,2,5,6,9,10,13,14,19)

22. Crea una función que reciba dos números primos p y q y devuelva $\phi(pq)$. Para agilizar el cálculo puedes obtener la fórmula general. (Grupo 1,5,9,13,19)

23. Crea una función que reciba tres enteros, a, n y m y devuelva el resto de dividir a^n entre m. Como si elevamos un entero a otro en general nos devuelve un entero que excede el tamaño máximo, usa para calcular el resto la función de Euler. (Grupo 2,6,10,14)

24. Crea una función que reciba dos primos y genere una clave pública y otra privada. Crea además una función que codifique/decodifique según se introuduzca la clave pública o la privada. Puedes ver el algoritmo en el enlace que aparece en la página web de la asignatura. (Opcional)

25. Crea una función que reciba las cifras de un número en base m (un vector o una lista con números entre 0 y $m-1$) y otra base n y devuelva el número en base n (puedes devolverlo como un vector o como una lista). (Grupo 3,7,11,15)

26. Crea una función que reciba m y n y devuelva el criterio de divisibilidad por m en base n (por ejemplo, los 10 primeros términos, o permitir que se elija el número de términos con otra variable). (Grupo 3,4,7,11,12,15,16)

27. Crea una función que reciba un número a en base n y devuelva si es divisible por m, llamando para ello a la función del apartado anterior para generar el criterio de divisibilidad y utilizando dicho criterio. (Grupo 4,8,12,16)

Normas de entrega:

1. Al finalizar la práctica, se entregarán únicamente los archivos fuentes, correspondientes a las funciones que se tenían que codificar y a un programa principal en el que se probarán las funciones.

2. En cada uno de los fuentes que se entreguen, se pondrá en la cabecera un comentario con el número del grupo y los nombres y apellidos de los integrantes.

3. Cada archivo debe estar convenientemente explicado mediante comentarios. Se valorará positivamente que se utilicen los resultados de la Teoría de Número para aumentar la eficiencia de los programas. En ese caso se debe incluir un comentario con el resultado que se haya utilizado.

4. Los archivos se guardarán en un directorio cuyo nombre será

   $$Grupo + ngrupo + inicialesapellidos.$$

   Por ejemplo, si vuestro grupo es el 12 y los apellidos son Álvarez, Fernández y Rodríguez, el directorio se llamará $grupo12afr$.

5. El directorio se comprimirá en un archivo con el mismo nombre y la extensión correspondiente al programa de compresión utilizada.

6. El archivo comprimido se enviará a $trinidad@unex.es$, antes de la fecha indicada en la página web de la asignatura.

7. Posteriormente, se anunciará una fecha para que uno de los miembros del grupo (elegido por el profesor el día de la exposición) explique alguno de los algoritmos desarrollados.