



Nombre y apellidos: \_\_\_\_\_

**Notas previas:**

- Escribe tu nombre y apellidos en esta hoja e inmediatamente en todas las que cojas, incluso las de sucio. El no hacerlo puede suponer tu expulsión.
- Responde de manera clara y ordenada. Utiliza una cara para cada respuesta. Puedes utilizar lápiz.
- “Recibe” es distinto de “lee del teclado”. “Devuelve” es distinto de “escribe” o “muestra en pantalla”.

1. **[1 punto]** Escribe en un recuadro lo que escribirá el siguiente programa en lenguaje C:

```
#include <stdio.h>
void main (void)
{
    int a=2, b=3, c=4;

    printf ("1. %d\n", ~a + 1);
    printf ("2. %d\n", a | b);
    printf ("3. %d\n", a ^ b);
    printf ("4. %d\n", c > b > a);
    printf ("5. %d\n", !a? b:c);
    printf ("6. %x\n", a * c + 2);
}
```

**Notas:**

- El operador < es asociativo a izquierdas
- El operador ~ obtiene el complemento (a 1) de un número
- El operador ^ realiza una operación o-exclusiva bit a bit
- Para números negativos se utiliza el complemento a 2
- El especificativo de formato %x muestra un número en formato hexadecimal en minúsculas
- Si es importante el tamaño de la palabra estima el que te parezca más adecuado

2. **[3 puntos]** Diseña el **diagrama de flujo** y **codifica** un programa en lenguaje C que lea dos números **naturales** y escriba en pantalla si son **“aceptables”**.

Decimos que dos números son aceptables cuando tienen al menos dos divisores comunes superiores a 1. Por ejemplo, el 6 y el 12 tienen el 2, el 3 y el 6 como divisores comunes.

El programa supondrá que los números introducidos son correctos.

A continuación se muestra el resultado de varias ejecuciones del programa solicitado con datos de entrada distintos (en **negrita** lo que introduce el usuario):

Primer número: <b>6</b> Segundo número: <b>12</b> El 6 y el 12 son aceptables
Primer número: <b>6</b> Segundo número: <b>6</b> El 6 y el 6 son aceptables
Primer número: <b>24</b> Segundo número: <b>27</b> El 24 y el 27 no son aceptables



3. [1 punto] Diseña el **diagrama de flujo de la cabecera**, incluyendo *nombre*, parámetros de *entrada y salida* y *valor devuelto*, y el **prototipo** en el lenguaje C para las siguientes especificaciones:

- Función `Pol12` que recibe los tres coeficientes de una ecuación de segundo grado ( $a$ ,  $b$  y  $c$ ) y calcula las dos raíces,  $x_1$  y  $x_2$ , devolviéndolas como parámetros por referencia. La función devuelve un 2, un 1 ó un 0 según sea una ecuación de 2º grado, de 1er grado o no es una ecuación, respectivamente.
- Función `SonOk` que recibe dos números,  $n_1$  y  $n_2$ , y devuelve un booleano que dice si los números son válidos para un cálculo que desconocemos (nos especificarán en un futuro).
- Función `EscribeDivisores` que recibe un número  $n$  y escribe en pantalla todos los números divisores de  $n$ .
- Función `SumAle2` que obtiene dos números aleatorios (mediante una función estándar, por ejemplo, `rand ( )`) y devuelve su suma.

**Nota importante:** no es necesario diseñar ni codificar las funciones.

4. [2 puntos] Podemos definir una serie geométrica a partir del primer elemento  $a_0$  y la razón  $r$  de manera que cada elemento  $a_i$  para  $i > 0$  se calcula a partir del anterior  $a_i = r \cdot a_{i-1}$ . Ejemplo:

$$a_0 = 4, r = 3, a_1 = 4 \cdot 3 = 12, a_2 = 12 \cdot 3 = 36, a_3 = 36 \cdot 3 = 108, \dots$$

**Codifica** un programa C que pida el primer elemento  $a_0$  y la razón  $r$  de una serie geométrica y luego pida un número natural  $x$  para mostrar en pantalla si pertenece o no a la serie. Supón que todos los datos introducidos son números naturales positivos y  $r > 1$ .

5. [3 puntos] (Elige entre esta pregunta o la siguiente) Sabemos que un número es divisible por 9 cuando sumando sucesivamente todos sus dígitos obtenemos el 9.

Codifica una **función** en lenguaje C que reciba un número natural **correcto** como una **cadena de caracteres** (como mucho de 1024 dígitos) y verifique si el número representado es **divisible por nueve**, devolviendo un valor booleano. Se utilizará el siguiente algoritmo: suma todos los dígitos dos a dos y forma una cadena con los dígitos resultantes hasta obtener una cadena de un solo dígito.

Ejemplo: el número "892125239" ¿es divisible por 9?

- "892125239" podemos verlo como: "89 | 21 | 25 | 23 | 9"
  - o  $8+9 \mid 2+1 \mid 2+5 \mid 2+3 \mid 9$
  - o  $17 \mid 3 \mid 7 \mid 5 \mid 9 \rightarrow "173759"$
- "173759"  $\rightarrow "17 \mid 37 \mid 59" \rightarrow 1+7 \mid 3+7 \mid 5+9 \rightarrow 8 \mid 10 \mid 14 \rightarrow "81014"$
- "81014"  $\rightarrow 8+1 \mid 0+1 \mid 4 \rightarrow 9 \mid 1 \mid 4 \rightarrow "914"$
- "914"  $\rightarrow 9+1 \mid 4 \rightarrow 10 \mid 4 \rightarrow "104"$
- "104"  $\rightarrow 1+0 \mid 4 \rightarrow 1 \mid 4 \rightarrow "14"$
- "14"  $\rightarrow 1+4 \rightarrow 5 \rightarrow "5"$

**Conclusión: Falso (no es divisible por 9)**



6. [3 puntos] (Elige entre esta pregunta o la anterior) Disponemos de las funciones:

DiaSis	Obtiene <b>día, mes y año</b> del reloj del sistema (de hoy)
DiaJul	Devuelve una <b>fecha</b> en formato numérico (juliano)
DiaGrg	Convierte de formato juliano a formato día-mes-año ( <b>gregoriano</b> )
DiaSem	Devuelve el <b>día de la semana</b> (0 - 6) de una fecha juliana
TxtMes	Devuelve el <b>texto de un mes</b> , numerado de 1 a 12

Sus prototipos, especificados en el fichero “**fechas.h**”, son los siguientes:

```
void DiaSis (int *dd, int *mm, int *aa);  
long DiaJul (int dd, int mm, int aa);  
void DiaGrg (long jul, int *dd, int *mm, int *aa);  
int DiaSem (long jul);
```

Un amigo supersticioso nos pide que codifiquemos:

- a. Un **programa** que pida un año y calcule qué fechas de ese año son 13 y martes. Un ejemplo de ejecución sería:

```
Introduce el año: 2009           (introducido por el usuario)  
Fechas martes-13 en 2009:  
13/01/2009  
13/10/2009
```

- b. Un **programa** que calcule cuándo van a caer los próximos 5 días que sean 13 y martes (utilizando como referencia la fecha del reloj del sistema). Un ejemplo de ejecución sería:

```
Hoy es 30/08/2007  
Sigüientes fechas martes-13:  
13/11/2007  
13/05/2008  
13/01/2009  
13/10/2009  
13/04/2010
```