

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

```
#include <stdio.h>
void main (void)
{
    int a=1, 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", a < b < b);
    printf ("5. %d\n", a? b:c);
    printf ("6. %x\n", b * c);
}
```

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 calcule la **media aritmética** y la **media normalizada** de una lista de calificaciones académicas (cada profesor define su rango de notas, por ejemplo, de 0 a 10, de 0 a 20 o de 0 a 100, nunca negativas).

Se denomina **media normalizada** (es un concepto utilizado en el centro para el que estamos desarrollando el programa) a la *media aritmética de todas las notas exceptuando la mayor y la menor*, por lo que no tiene sentido si tenemos menos de tres notas.

Al ejecutar el programa el usuario introducirá las notas **finalizando con -1**.

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

Introduce nota (para finalizar -1): -1 No se ha introducido ninguna nota
Introduce nota (para finalizar -1): 8 Introduce nota (para finalizar -1): 6 Introduce nota (para finalizar -1): -1 Media: 7.00 No hay suficientes notas para la media normalizada
Introduce nota (para finalizar -1): 1 Introduce nota (para finalizar -1): 6 Introduce nota (para finalizar -1): 6.5 Introduce nota (para finalizar -1): 7 Introduce nota (para finalizar -1): -1 Media: 5.12 Media normalizada: 6.25

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 siguiente funciones:
- Función CalMax que recibe tres números enteros y devuelve el mayor de los tres.
 - Función Desplaza que recibe tres variables x1, x2 y x3 e intercambia sus valores, de manera que x2 reciba el valor de x1, x3 reciba el valor de x2 y x1 reciba el valor de x3.
 - Función EscribeSerie que recibe el valor del orden n y escribe en pantalla todos los números de la serie de Fibonacci de 0 a n.
 - Función LeeOpcion que escribe en pantalla un texto, pide al usuario que elija una opción y devuelve la opción introducida cuando ésta sea correcta.
4. [2 puntos] Un número es **perfecto** si la suma de todos sus divisores excluyendo a sí mismo es igual al mismo número.
- Por ejemplo, el 6 es perfecto ya que $1+2+3 = 6$.
- Codifica una función que calcule si un número es perfecto.
 - Codifica un programa que lea un número y escriba en pantalla si es o no perfecto invocando a la función del apartado anterior.
5. [3 puntos] (**Elige** entre esta pregunta o la siguiente) Disponemos de las funciones:

DiaSis	Devuelve día, mes y año del reloj del sistema (de hoy)
DiaJul	Devuelve una fecha en formato numérico (juliano)
DiaGrg	Convierte de formato juliano a formato día-mes-año (gregoriano)
DiaSem	Devuelve el día de la semana (0 - 6) de una fecha juliana
TxtMes	Devuelve el texto de un mes , 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);
char *TxtMes (int mes);
```

Escribe una **función** que escriba el mes actual (según la fecha del sistema) de la siguiente manera (no importa si no es eficiente):

2007 febrero						
L	M	M	J	V	S	D
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

6. [3 puntos] (Elige entre esta pregunta o la anterior) Codifica una **función** en lenguaje C que reciba un número natural `num` (no lo verificará) y una base `bas` entre 2 y 24, calculando la cadena correspondiente al número `num` en la base `bas`. Para dígitos entre 10 y 23 utilícen las letras en mayúscula de la 'A' a la 'N' en las bases que lo requieran. Se propone el siguiente prototipo:

```
void NumEnBase (int num, int bas, char *numstr);
```

Codifica un **programa** que pida y lea un número y una base, obtenga la cadena correspondiente mediante la función `NumEnBase` y la escriba en pantalla.

Ejemplos:

```
Escribe un número: 36
Escribe una base: 19
36 en base 19: 1H
```

```
Escribe un número: 36
Escribe una base: 2
36 en base 2: 100100
```