1. Escriba un probrama en C por cada apartado siguiente: NOTA: un ejemplo de salida o impresión de la ejecución del podría ser:

   ```
   Este programa le permite calcular el area de un circulo dado su radio.
   Introduzca el valor del radio: 5
   El valor del area del ciruculo es: 78.525
   ```

   a) Cálculo de área de un círculo.

   b) Cálculo de circunferencia de un círculo.

   c) Cálculo de la media de tres números.

   d) Cálculo de los días, horas, minutos y segundos dada una cantidad en segundos. El programa deberá tener una salida o impresión como la siguiente:
   ```
   Introduzca el número de segundos: 93662
   93662 segundo(s) son:
   - 1 dia(s)
   - 2 hora(s)
   - 1 minuto(s)
   - 2 segundo(s)
   ```
   Considere el uso de constantes en este ejercicio.

   e) Cálculo de la velocidad de un móvil dado el espacio que ha recorrido y el tiempo que ha tardado. Considere que el movimiento es rectilíneo uniforme.

   f) Cálculo del número de semanas completas que han transcurrido dado un número de días.

   g) Cálculo del perímetro de un polígono regular dado el número de lados y la longitud del lado.

   h) Cálculo del área de un polígono regular dado el número de lados, la longitud del lado y la apotema.

   i) Cálculo de números binarios que se pueden representar dado un número de bits.

   j) Cálculo del número de Mega Bytes, Kilo Bytes y Bytes que ocupa una determinada cantidad de Bytes. El programa deberá tener una salida o impresión como la siguiente:
   ```
   Introduzca el número de Bytes: 1100000
   1100000 Bytes(s) ocupa(n):
   - 1 Mega Byte(s)
   - 50 Kilo Byte(s)
   - 224 Byte(s)
   ```

   k) Intercambio del valor de dos variables. El valor de las variables debe darlas el usuario.

   l) Cálculo del valor de un polinomio de segundo grado $a_2x^2 + a_1x + a_0$, considerando que los coeficientes son constantes (elija el valor que desee) y dado un valor de x que se solicitará al usuario.

   m) Repita el ejercicio $1l$ pero el valor de los coeficientes también serán aportados por el usuario.

   n) Repita el ejercicio $1m$ pero almacene los valores de los coeficientes en un array.

   ñ) Repita el ejercicio $1c$ pero almacene los valores en un array de tres elementos.

o) Repita el ejercicio 1*d* pero almacene los días, horas, minutos y segundos resultantes en un array.

p) Repita el ejercicio 1*j* pero almacene los Mega Bytes, Kilo bytes y Bytes resultantes en un array.

q) Repita el ejercicio 1*g* pero almacene el numero de lados, la longitud del lado y el perímetro en un array.

r) Repita el ejercicio 1*k* pero almacene los valores a intercambiar en un array.

2. La letra del DNI se obtiene calculando el resto de la división del número del DNI entre el número 23. Una vez obtenido el resto se asigna la letra en función de la siguiente tabla:

Resto	0	1	2	3	4	5	6	7	8	9	10	11
Letra	T	R	W	A	G	M	Y	F	P	D	X	B
Resto	12	13	14	15	16	17	18	19	20	21	22	
Letra	N	J	Z	S	Q	V	H	L	C	K	E	

Desarrolle un programa que solicite al usuario el número de DNI y le informe de la letra que correspondería para ese número. La salida del programa debe tener una estructura como la siguiente:

```
Este programa determina la letra del DNI
Introduzca el numero del DNI: 46
Al DNI 46 le corresponde la letra 0
```

Valide que el DNI introducido es mayor que 0 y menor que 99 999 999

Tenga en cuenta al realizar este programa en C que el tipo de dato que selecciones para guardar el DNI debe permitir valores de hasta 99 999 999

La solución debe realizar el menor número posible de comparaciones, así que seleccione el esquema condicional más adecuado.

3. Un año es bisiesto cuando es divisible entre 4 (el año 2012 fue bisiesto), sin embargo, los años divisibles entre 100 no lo son (el año 1 900 no fue bisiesto), pero sí que lo son los años divisibles entre 400 (el año 2 000 sí fue bisiesto). Escriba un programa que dado un año a, proporcionado por el usuario, informe si es bisiesto o no. Realice este ejercicio del tal forma que la expresión lógica de un único condicional, compruebe si el año es bisiesto. Repita el ejercicio de tal forma que la expresión lógica de un único condicional compruebe que el año no es bisiesto. Valide que el año introducido sea mayor que 0 y menor que 5 000.

4. Desarrolle un algoritmo que dado un número entero positivo, n, muestre por pantalla los n primeros términos de la sucesión de Fibonacci: $0, 1, 1, 2, 3, 5, 8, \ldots$. n debe ser menor o igual a 50.

5. Desarrolle un algoritmo que calcule la raíz cuadrada r entera más próxima a un número dado n. n será proporcionado por el usuario y estará dentro del intervalo [0,4 000]. Además $r^2 \leq n$.

6. Modifique el ejercicio 5 para que pueda aproximar la raíz del número con $1, 2, 3, \ldots, d$ decimales, siendo d un número entre 1 y 8 que proporcione el usuario. El procedimiento debe acercarse a la solución de forma progresiva, por ejemplo: para el número 12 debe indicar:

```
Con 0 cifras decimales la raiz mas proxima es 3
Con 1 cifras decimales la raiz mas proxima es 3.4
Con 2 cifras decimales la raiz mas proxima es 3.46
Con 3 cifras decimales la raiz mas proxima es 3.464
...
```

7. Cree una función para cada una de las siguientes funciones:

$$a)\ f_1(x) = \begin{cases} x+1, & x < 0 \\ -x, & x > 0 \end{cases} \qquad c)\ f_3(x) = \begin{cases} cos(x), & x < -\pi/2 \\ -sin(x), & x > \pi/2 \end{cases}$$

$$b)\ f_2(x) = \begin{cases} 2x+4, & x \in [0, 10) \\ 3x^2 + 5x - 1, & x \in [10, 20) \end{cases} \qquad d)\ f_4(x) = \begin{cases} x^3, & x \leq 0 \\ \sqrt{x}, & x \geq 0 \end{cases}$$

$$e)\ f_5(x, y) = x^2 + y^2$$

   Cada función deberá tener un parámetro numérico x (e y si corresponde) y devolverá un valor numérico que será la evaluación de la función. Si el parámetro x (o y) es un valor para el que la función no está definida, la función devolverá el valor -1.

   Para las funciones de los apartados $7a$, $7b$, $7d$ y $7d$. Cree un programa principal `main` para cada uno de los algoritmos con nombre desarrollados que haga las siguientes tareas.

   - Pida un intervalo válido $[a_0, b_0]$ por teclado al usuario.
   - Pida un valor δ, $\delta \in [10^{-5}, 1]$.
   - Muestre el por pantalla el valor de la función recorriendo todos los posibles valores del intervalo según indique el incremento δ: $a_0, a_0 + \delta, a_0 + 2\delta, \ldots$. Tenga en cuenta que el último valor de esta secuencia tendrá que ser $\leq b_0$, pero en ningún caso podrá ser $> b_0$.

   Un ejemplo de salida sería

```
Introduzca un intervalo valido:
  Valor de a0: -0.2
  Valor de b0: 0.2
Introduzca el valor de delta [1e-5,1]: 0.1
Evaluando f1 en el intervalo [-0.2, 0.2] con delta=0.1
f1(-0.2): 0.8
f1(-0.1): 0.9
f1(0.0): -1
f1(0.1): -0.1
f1(0.2): -0.2
```

8. Repita el ejercicio 7 pero las funciones recibirán dos parámetros: el valor numérico x y un parámetro por referencia que tomará el valor de $f(x)$ si f está definida para el x dado, o

`-1`, en caso contrario. Además la función devolverá el valor lógico `true` si la función está definida en el x dado o `false`, en caso contrario.

Durante el recorrido de los valores del intervalo, el programa principal detectará si la función está definida en cada punto. Si lo está se mostrará el valor de la función, y si no, el mensaje: `no definida`. Un ejemplo de salida sería:

```
Introduzca un intervalo valido:
  Valor de a0: -0.2
  Valor de b0: 0.2
Introduzca el valor de delta [1e-5,1]: 0.1
Evaluando f1 en el intervalo [-0.2, 0.2] con delta=0.1
f1(-0.2): 0.8
f1(-0.1): 0.9
f1(0.0): no definida
f1(0.1): -0.1
f1(0.2): -0.2
```

9. Cree una función que permita calcular la suma de los elementos de un *array* de tipo `double`, teniendo en cuenta que ha de pasar el tamaño del *array*. El valor de la suma ha de retornarse usando `return`.

10. Diseñe una estructura de datos que contenta los valores de un *array* de `double`, el número de elementos del *array* y la suma de sus elementos. Asigne un nombre a la nueva estructura de datos con typedef y modifique la función del ejercicio 9 para que reciba un único parámetro y calcule la suma. La nueva función no deberá retornar nada mediante `return`.

Los siguientes ejercicios permiten ver ejemplos de los distintos conceptos presentados en las sesiones de prácticas. Como trabajo autónomo, codifique, entienda y responda a las preguntas que se formulan en alguno de ellos.

11. Uso de la función `printf` con enteros:

```c
#include <stdio.h>
// Uso del printf con enteros
int main(void)
{
   int a=5;
   int b=6;
   int c=44;

   printf("La suma de %d + %d es %d\n",a,b,a+b);
   printf("%d en octal es %o y en hexadecimal %x\n",c,c,c);
   return 0;
}
```

12. Uso de la función `printf` con reales:

```c
#include <stdio.h>
//uso del printf con reales
int main(void)
{
   float a=5.1;
   float b=4.9;
   float c=0.0001;
   printf("La suma de %f + %f es %f\n",a,b,a+b);
```

```
  printf("El valor de la variable a con 3 decimales es %.3f\n",a);
  printf("La opcion g tambien es util %g\n",a);
  return 0;
}
```

13. Uso de la función `scanf`:

```
#include <stdio.h>
//ejemplo del uso del scanf
int main(void)
{
   int a,b;
   float c;
   printf("Introduzca un numero entero: ");
   scanf("%d",&a);
   printf("Introdujo el numero %d\n",a);

   //Podemos leer varios numeros
   printf("Introduce dos enteros separados por un espacio: ");
   scanf("%d %d",&a,&b);
   printf("Ha introducido los numeros %d y %d\n",a,b);

   //Podemos leer un numero entero y un real
   printf("Introduce un entero y un real separados por un espacio: ");
   scanf("%d %f",&a,&c);
   printf("Ha introducido los numeros %d y %f\n",a,c);

   /*Para leer strings existe una funcion llamada fgets. scanf puede leer
   strings pero no si contienen espacios.*/
   return 0;
}
```

14. Ejemplo de declaración y uso de vectores.

```
#include <stdio.h>
//ejemplo del uso de vectores
#define TAM_VECTOR 5 //Declaracion de una constante!
int main(void)
{
   float v[TAM_VECTOR];
   for (int i=0; i<TAM_VECTOR; i++)
   {
      v[i] = i*2;
   }

   for (int i=0; i<TAM_VECTOR; i++)
   {
      printf("v[%d] = %g\n",i,v[i]);
   }
   return 0;
}
```

15. Ejemplo con cadena de caracteres.

```
#include<stdio.h>
#include<stdlib.h>
```

```c
#define TAM_MAX 200

int stringSize(char * v);

int main(void)
{
   char v[] = "Hola mundo";
   char w[TAM_MAX];
   int strSize;

   strSize = stringSize(v);
   printf("El string \"%s\" tiene %d caracteres.\n",v,strSize);

   printf("Escribe algo: ");
   fgets(w,TAM_MAX,stdin);
   strSize = stringSize(w);
   printf("El string \"%s\" tiene %d caracteres.\n",w,strSize);

   return 0;
}

int stringSize(char *v) //Escribir int stringSize(char []) seria
   equivalente
{
   int size = 0;
   for (int i=0; v[i]!= 0; i++)
   {
      size++;
   }
   return size;
}
```

¿Por qué antes de la declaración de la función `main` se ha escrito `int stringSize(char * v);`? ¿Por qué en la fucnicón que determina el tamaño de un string se cuenta hasta la aparición del caracter `0` (que no `'0'`)?
El string `w` se ha declarado con un tamaño de `200`, sin embargo la función que determina el tamaño de dicho string no dice siempre que ese tamaño sea 200 ¿Por qué?.
Documéntese sobre la función `atoi` ¿para que sirve?

16. Ejemplo del operador `&`.

```c
#include <stdio.h>
//Ejemplo del operador &
int main(void)
{
   int b = 5;
   printf("La variable b coniene el valor %d y esta en la direccion de
   memoria %p\n",b,&b);
   // %p permite imprimir una direccion de memoria (puntero)
   return 0;
}
```

¿Cómo determinaría lo que ocupa un variable de tipo puntero en memoria?

17. Punteros: el operador `*`

```c
#include <stdio.h>
//Primer ejemplo de punteros
```

```c
int main(void)
{
   int b=55;
   int* c;
   c=NULL;
   c = &b; //Se guarda en c la direccion de memoria donde esta almacenada
   la variable b
   printf("La variable b contiene el valor %d y esta en la direccion de
   memoria %p\n",b,&b);

   //Los punteros contienen direcciones de memoria
   printf("La variable c es un puntero que guarda la direccion de memoria
   %p\n",c);

   //Podemos mostrar el valor apuntado por un puntero (valor contenido en
   la direccion de memoria que almacena)
   printf("La variable c es un puntero que apunta al valor: %d\n",*c);
   return 0;
}
```

¿Por qué el contenido de `c` es la dirección de memoria de la variable `b`?
¿Cuál es el puntero `NULL`? ¿Por qué cree que es buena idea inicializar los puteros a `NULL`?

18. Punteros: el operador `*`

```c
#include <stdio.h>
//ejemplo de uso de punteros
int main(void)
{
   int a=55;
   int* b;

   b = &a;
   printf("El valor de a es %d\n",a);
   *b = *b + 10;
   printf("Ahora el valor de a es %d\n",a);
   return 0;
}
```

¿Por qué cambia el valor de la variable `a`?

19. Punteros no inicializados

```c
#include <stdio.h>
//ejemplo de uso de un puntero no inicializado
int main(void)
{
   int* a;
   int b=2;
   int c;
   c = b + *a; //Esta expresion genera un error en tiempo de ejecucion
   return 0;
}
```

¿Por qué se genera un error en tiempo de ejecución? ¿Cómo lo solucionaría?

20. Paso de parámetros por referencia

```c
#include <stdio.h>

void suma10P(int* a);
void suma10(int a);

int main(void)
{
   int a1=10;
   suma10P(&a1);
   printf("El valor de a despues de llamar a suma10P es %d\n",a1);
   a1=10;
   suma10(a1);
   printf("El valor de a despues de llamar a suma10 es %d\n",a1);
   return 0;
}

void suma10P(int* a)
{
   *a = *a + 10;
}

void suma10(int a)
{
   a = a + 10;
}
```

¿Cuál de las dos funciones modifica el valor del parámetro? ¿Por qué?