Ejercicios de repaso

Qué imprime?

```
float i;
for (i = 1.28; i < 16; i*=4)
    printf("%.1f ", i);
```

Qué imprime?

```
int i, a=0;
for (i = 0 ;  i==100 ;  i++)
      a += 3;
printf("el valor de a es %d ",a);
```

a. El valor de a es 300.
b. El valor de a es 0.
c. El valor de a es 100.
d. El valor de a es 303.
e. No imprime nada dado que el código presenta errores en la sintaxis del *for* al compilar.
f. No imprime nada dado que queda en un bucle infinito.

Qué imprime?

```
int i;
for (i = 5; i > 0; i-=2)
    printf("%d ", !(i % 3));
```

Qué imprime?

```
int i;
for (i = 5; i > 0; i=-2)
    printf("%d ", !(i % 3));
```

Qué imprime?

```
int i;
for (i = 1; i < 4; i++){
  if (i==2)
    continue;
  printf("%d ", i);
}
```

Qué imprime?

```
int i, a;
for (i = 9; i ; i/=3) {
    a = i-3 ? ++i : i++ ;
    printf("%d ", a );
}
```

Qué imprime?

```
int i;
for (i=0 ; i<3 ; ++i) {
      static int s = 4;
      if (--s % 3)
         printf("s = %d\n", s--);
}
```

Resuelva

- Escriba cuatro instrucciones diferentes de C que sumen 1 a la variable entera x.

Corrija el error

```
float y;

for ( y = .1; y != 1.0; y += .1 )
   printf( "%f \n", y );
```

Indique la opción verdadera

a. Un programa puede compilar con *errores* pero no con *warnings*.
b. Un programa puede compilar con *warnings* y *errores*.
c. Un programa puede compilar con *warnings* pero no puede ejecutarse.
d. Un programa con *warnings* puede ejecutarse pero podrían aparecer resultados inesperados.
e. Los *warnings* son errores críticos.

Para cada inciso indique si es verdadero (V) o falso (F)

	La siguiente instrucción no compila: `printf("%d\n", !4);`
	Los operadores aritméticos *, /, %, + y – tienen el mismo nivel de precedencia.
	El operador módulo (%) puede utilizarse sólo con operandos enteros.

Para cada inciso indique si es verdadero (V) o falso (F)

	No es posible asignar ningún valor entero a una variable de tipo puntero.
	No es posible comparar dos variables estructuras aunque sean del mismo tipo.
	Un puntero *void* puede asignarse o recibir valor de cualquier tipo de puntero.

Para cada inciso indique si es verdadero (V) o falso (F)

	Si una función recibe como parámetro int * const Ptr no podrá modificar lo apuntado por Ptr.
	Si se imprime una variable char (toma valores entre -128 y 127) inmediatamente después de asignarle el valor 128 se visualizará el valor -128.
	Dos estructuras distintas no pueden tener campos con el mismo nombre.

Complete las funciones

```
void sumaColumnas(const int * const, int, int, int * const);
void mostrar(const int [], int);

int main()
{   enum {N=3, M=2};
    int Mat[N][M] = {10,20,30,40,50,60};
    int sumas[M];

    sumaColumnas(Mat, N, M, sumas);
    mostrar(sumas, M);

    return 0;
}
```

Qué imprime?

```
int a=5, b=6;
printf("a & b = %d", a & b);
```

Qué imprime?

```
int c=2, d=7;
printf("c && d = %d", c && d);
```

Qué imprime?

```
enum {UNO, DOS, TRES=0, CUATRO} p;
int suma=0, V[]={1,2,3,4,5,6,7,8,9};

for (p=UNO; p<CUATRO; p++)
     suma = suma + V[p];

printf("suma = %d", suma);
```

Qué imprime?

y si sacamos “=0”?

```
enum {UNO, DOS, TRES  , CUATRO} p;
int suma=0, V[]={1,2,3,4,5,6,7,8,9};

for (p=UNO; p<CUATRO; p++)
     suma = suma + V[p];

printf("suma = %d", suma);
```

La función multiplo determina si el valor recibido como parámetro es múltiplo de X o no

¿Cuánto vale X?

```c
/* determina si num es un múltiplo de X */
int multiplo( int num )
{
    int i;
    int mascara = 1;
    int mult = 1;

    for ( i = 1; i <= 10; i++, mascara <<= 1 ) {
          if ( ( num & mascara ) != 0 ) {
                mult = 0;
                break;
          }
    }
    return mult;
}
```

Ejer_FuncionMultiplo.c

```c
int misterio( unsigned bits )
{
    unsigned i;
    unsigned mascara = 1 << 31;
    unsigned total = 0;

    for ( i = 1; i <= 32; i++, bits <<= 1 )

        if ( ( bits & mascara ) == mascara )
            total++;

    return !( total % 2 ) ? 1 : 0;
}
```

¿Qué retorna la función misterio?

Ejer_FuncionMisterio.c

Identifique los errores

```c
#include <stdio.h>
#include <string.h>
void invertir(char *);
int main()
{    char linea[50];

     printf("Ingrese una linea: ");
     gets("%s", linea);

     puts("%s", linea);
     invertir(linea);
     puts("%s", linea);

     return 0;
}
```

```c
void invertir(char * L);
{   int i, N = sizeof(L);
    char aux;
    for (i=0; i<N/2; i++){
        aux = L[i];
        L[i] = L[N-i];
        L[N-i] = aux;
    }
}
```

Utilizando la sigte definición

```
struct alu {
    char apellido[50];
    char nombre[50];
    char legajo[8];
};
```

a) Renombre el tipo **struct alu** a **alumno**.

b) Defina una función que permita inicializar una estructura **alumno**.

c) Defina un arreglo de 10 elementos de tipo alumno e inicialice cada uno de ellos utilizando la función definida en el punto b).

d) Imprima la información de cada alumno con el siguiente formato:

Apellido y nombre: Pérez, Juan | Legajo: 7751/8

Apellido y nombre: García, Pablo | Legajo: 6952/1