1. Analice el siguiente código e indique cuál es la opción correcta:

```c
#include <stdio.h>

int main(){
    int a, cant = 0;
    scanf("%d",&a);
    while (a % 3) {
            ++cant;
            scanf("%d",&a);
    }
    return 0;
}

```

   a. El programa almacena en *cant* la cantidad de números leídos hasta ingresar el 3.
   b. El programa almacena en *cant* la cantidad de números leídos consecutivamente que no son múltiplos de 3.
   c. El programa almacena en *cant* la cantidad de números leídos consecutivamente que son múltiplos de 3.
   d. El programa no compila.

2. Indique qué imprime el siguiente código:

```c
#include <stdio.h>

int main(){
    int i, a= 0;

    for (i=0; i==100; i++)
        a+= 3;

    printf("El valor de a es %d",a);

    return 0;
}

```

   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.

3. Indique la opción correcta si a partir del siguiente código se desea cambiar el valor de la variable *c* a 15.

```c
#include <stdio.h>

int main(){

    int *a, *b, c, d, *e;
    c = 10;
    d = c*2;
    a = e;
    e = &c;
    b = a = e;

    /* La instrucción elegida va aquí */

    return 0;
}

```

   a. e = 15;
   b. *e = 15;
   c. *c = 15;
   d. a = 15;
   e. &a = 15;
   f. b = 15;
   g. *b = 15;
   h. Ninguna de las opciones anteriores cambian el valor de *c* a 15.

5. Indique cuál de las siguientes opciones es 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.

6. Indique cuál de las siguientes opciones sobre punteros es falsa:
   a. Contienen una dirección de memoria.
   b. Permiten simular el pasaje por referencia.
   c. Al aplicar el operando * (de indirección) sobre una variable puntero se obtiene la dirección de memoria apuntada por el puntero.
   d. Existen 3 valores posibles con los cuales pueden ser inicializados.
   e. Al aplicar el operando & (de dirección) sobre una variable puntero se obtiene la dirección de memoria donde se encuentra la variable puntero.

7. El siguiente bloque de código define la estructura `struct alu`:

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

   A partir de la definición anterior, realice:
   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: Perez, Juan | Legajo: 7751/8
      Apellido y nombre: García, Pablo | Legajo: 6952/1
      …

8. Escriba un programa que lea palabras ingresadas por teclado hasta la lectura de la palabra “FIN” (las palabras tienen como máximo 20 caracteres). Al finalizar, informe cual es la palabra de menor longitud.

9. a) Defina un tipo enumerativo para trabajar con tres tipos de figura: cuadrado, triángulo rectángulo y círculo. Luego defina una estructura para representar una figura utilizando el enumerativo junto con las siguientes características. Todas las figuras tienen el atributo color. Además, para la figura cuadrado se indica la longitud de su lado, para el triángulo rectángulo la longitud de los lados que forman el ángulo recto y para el círculo su radio.

   b) Escriba una función que reciba una estructura definida en a) e imprima: el nombre de la figura, su color y su área.