Cuestionario Lenguaje C: Punteros. Memoria dinámica.

1) ¿Qué es un puntero en lenguaje C? ¿Cómo se declara? ¿Cómo se le asigna un valor?
2) ¿Por qué se dice que un puntero desreferenciado es un alias de la variable apuntada? ¿Cómo se accede a la variable apuntada, a través del puntero?
3) ¿Cuáles son las operaciones válidas con el tipo puntero?
4) Si todo puntero guarda una dirección de memoria y todas las direcciones de memoria son del mismo tipo, ¿por qué es necesario indicar en la declaración del puntero el tipo de datos al que apuntará?
5) ¿Cuánto se modifica la dirección de memoria a la que apunta el puntero, si le sumamos 1 al puntero, cuando está apuntando a una variable de tipo float? Compruébenlo.
6) ¿Qué es un puntero genérico y para qué se usa?
7) ¿Qué es un puntero nulo y para qué se usa?
8) Diferencien las declaraciones:
   ```
   const char *pc;
   char *const pc;
   const char  *const pc;
   ```
   indique usos válidos y erróneos de cada una de ellas.
9) ¿Cuál es la relación entre los punteros y los arrays? ¿Cuál es la diferencia? ¿Por qué las siguientes expresiones son equivalentes?:
   ```
   lista[ind], *(lista+ind), plistafind], *(plista+ind)
   ```
   ¿A cuál de ellas llamamos notación de array e índice? ¿cuál llamamos notación de puntero y desplazamiento? Esas serán las que emplearemos en los ejercicios.
10) Expliquen por qué las funciones de la librería string.h reciben parámetros de tipo char *. ¿Qué relación hay entre punteros y strings?
11) En el siguiente código hay varias asignaciones a un string: ¿Cuál de ellas no es válida y porqué?¿Cómo soluciona ese error?
    ```
    char *nombre =  "Ceballos Sierra";
    char nombre1[] =  "Ceballos Sierra";
    printf  ("%s\n",  nombre);
    nombre =  "Horacio";
    printf  ("%s\n",  nombre);
    nonbre1 =  "Horacio";
    printf  ("%s",  nombre1);
    ```
12) Diferencien array de strings, de array de punteros a strings. ¿Qué ventajas tiene el uso de éste último sobre el primero? ¿Cómo se declara cada tipo?
13) Diferencien asignación estática y dinámica de memoria. ¿En qué bloques de memoria alojan las variables cada tipo de asignación? ¿Cómo y cuándo se gestionan altas y bajas de variables en estos bloques de memoria?
14) ¿Cuáles son las funciones de C++ para manejo de memoria dinámica? Indiquen cómo se emplean para asignar memoria para arreglos dinámicos numéricos y de strings.
15) ¿Cómo se relaciona el tipo matriz (o sea array de dos dimensiones) con el tipo puntero a puntero. ¿Cómo se declara cada tipo? Explique la notación de puntero y desplazamiento para acceder a un elemento de la matriz cuando se ha declarado como un puntero a puntero.
16) ¿Cómo se declara un puntero a un struct? ¿Cómo se le hace una asignación válida? ¿Cómo se accede a los campos del struct a través del puntero? ¿Por qué usar puntero a struct en lugar de usar struct directamente?
17) ¿A qué se denomina argumento formal y argumento real? Ejemplifiquen.
18) ¿Diferencien paso de argumentos a una función por valor y por referencia? ¿Qué relación hay entre argumentos reales y formales, en cada caso? Ejemplifiquen con una función que intercambie el valor de sus dos argumentos de tipo int.

19) ¿Por qué la función scanf requiere anteponer el & al nombre de la variable?
20) ¿Cómo pueden hacer para que una función retorne más de una variable (que no sean campos de un struct)?
21) ¿Por qué no es posible ni necesario que una función retorne un array?
22) ¿Por qué los arrays como argumento a una función son siempre pasados por referencia?
23) ¿Cuál es el problema que se presenta en una función que retorne (return) un puntero? ¿Cómo se soluciona? Ejemplifiquen.