**Teoría 1. Conteste brevemente las siguientes preguntas:**

**a. ¿Cómo se puede evitar que un parámetro pasado por referencia sea modificado?**

**b. ¿Qué diferencia existe entre una unión y una estructura?**

Una unión es similar a una struct, puede contener elementos de diferentes tipos y tamaños. La diferencia radica en que todos los campos comparten la mismamemoria, si se modifica el valor de uno de ellos se pierden los valores del resto.El tamaño (sizeof) de una union es igual al tamaño del mayor de sus miembros.

**c. ¿Qué relación existe entre las declaraciones int vector[] e int \*vector?**

La relación es que ambas declaraciones, justamente como la palabra nos indica declaran un array, en el primer caso de manera estatica y en el segundo caso de una manera dinámica, es decir que se le reserva un espacio especifico en memoria sin desperdiciar la misma, y que en el proceso de ejecución puede cambiar.

**d. ¿Cuál es la utilidad de un puntero a función?**

Un puntero a función es una variable que almacena la dirección de una función. Esta función puede ser llamada más tarde, a través del puntero. Este tipo de construcción es útil ya que encapsula comportamiento, que puede ser llamado a través de un puntero.

Los punteros a funciones también se usan para realizar funciones de retrollamada (callback, en inglés) donde se encarga de pasar como argumento a una función el nombre de otra función.

**e. ¿De qué manera está organizada la memoria en un sistema operativo? Detalle qué función cumple cada división y realice un diagrama.**

Para poder realizar la ejecución de un programa, el sistema operativo y la CPU adoptan un esquema de partición de memoria con funcionalidades distintas: **code**, **stack**, **data** y **heap**. Cuando se le pide al SO que ejecute un programa, éste carga el código ejecutable en **code**, y además, reserva zonas para **data** (variables globales y constantes),para el **stack** (llamado a funciones y variables locales), y para el **heap** (gestión dinámica de memoria).

**2. Identifique si las siguientes afirmaciones son verdaderas o falsas y justifique en el caso de ser falsa.**

**a. Las uniones se inicializan de la misma manera que las estructuras. FALSO**

Debido a que se almacena sólo un valor, las reglas para inicializar las uniones son diferentes que en el caso de las estructuras. Existen tres alternativas:

1. Copiar los elementos desde otra estructura del mismo tipo.

2. Asignarle un valor al primer elemento de la union.

3. En C99 se puede inicializar cualquier elemento indicándolo explícitamente**.**

**b. Las estructuras se pasan siempre a las funciones por referencia. FALSO**

Una estructura puede ser pasado por valor como por referencia, al igual que un dato simple.  
La diferencia de pasar un dato por valor o por referencia, es que por valor se hace una copia haciendo imposible que puedas modificar la estructura original. Todo cambio que les haga adentro de la funcion, no trendra ocurrencia en la original.  
En cambio, al pasar una estructura por referencia, lo que se pasa es un puntero a la estructura, y todo cambio que hagas dentro de tu funcion se haran sobre la estructura original.

**c. Los punteros de diferentes tipos no pueden ser asignados uno al otro. FALSO**

Ya que cuando se declara un puntero se reserva memoria para almacenar una dirección de memoria, pero no para **almacenar el dato al que apunta el puntero**. Esta reserva es independiente del tipo de dato ocupado (float, char, int, …).

**d. Las variables almacenadas en el stack son accesibles desde cualquier función. FALSO**

Ya que en el stack se almacenan los parámetros de las funciones (de entrada y retorno)y las **variables locales** utilizadas por cada una. No las variables globales, es decir que las variables almacenadas en el stack solo pueden ser llamadas dentro de esa misma función, para tener una variable que pueda ser accesible desde cualquier función, deben estar almacenadas en data y no en el stack.