Tarea 11:

Fecha de entrega: ***ANTES** del 29 de Octubre.*

Toda la tarea se debe de hacer usando memoria dinámica (`malloc()` y `free`()) cuando sea requerida memoria.

a) Hacer una función que recibe como parámetro un `unsigned int` (por ejemplo `unsigned int a = 255`), y que usa un ciclo para mandar a la pantalla la trama de bits que componen al número (por ejemplo 00000000000000000000000011111111). Para eso usar una máscara inicializada al primer bit como `unsigned int mask = 1`, y usar corrimientos a la izquierda para pasar cada vez el siguiente bit.

b) Hacer un programa que declara una variable `a` de tipo `double`, una `b` de tipo `int` y un arreglo `arr` de `N`=10 variables tipo `float`. Inicializar todas ellas a cero. Ahora, sin usar ninguna de las variables `a, b` ni `arr` en ninguna de las sigientes instrucciones: modificar todas las variables a un valor que tu quieras (usando variables de tipo apuntador) y mandar a pantalla su nuevo valor usando la función `printf()`.

c) Diseminar este chiste por el mundo :D (https://www.cimat.mx/~alram/elem_comp/chiste_cadenas.txt) (No hace falta reportar nada, je je).

d) Dado un arreglo de caracteres `char vc[24]` , llenarlo con las 24 primeras letras del lenguaje inglés (de la `'a'` a la `'x'`) en orden ascendente con un ciclo `for`. Posteriormente, poner un apuntador `aptrLI` de tipo `unsigned int` al inicio del vector `vc` (quizá necesites hacer un cast entre tipos de apuntadores), imprimir en pantall todos los `unsigned int` que caben en ese espacio de memoria. Explicar por qué salen los números que salen en pantalla.

e) Hacer una función que recibe como parámetros, un vector de enteros y apuntadores a las variables de tipo `float: promedio, varianza, moda, maximo, minimo.` La funcion calcula internamente esos valores y los deja en las direcciones de los apuntadores.

f) Hacer una función que recibe como parámetro un apuntador a una matriz cuadrada (la matriz debe de ser definida como estructura, conteniendo `nRens, nCols` y el arreglo 2D), y deja en ella misma su transpuesta. Usar -> en su código. Usar para todo memoria dinámica.

g) Hacer que un apuntador `unsigned int` apunte a un float usando **casting** entre apuntadores (preguntar de esto en la siguiente clase), y, ahora que ya tienes el pedazo de memoria como un `unsigned int` usando mascaras de bits, verificar que un valor `float` 0.15625 se almacena en bits como se muestra en https://www.cimat.mx/~alram/elem_comp/float_en_bits.jpg

(no lo necesitas para este curso, pero si quieres saber más, el dibujo fue tomado de https://es.wikipedia.org/wiki/Formato_en_coma_flotante_de_simple_precisión).

h) Declarar el nuevo tipo de dato `Matrix`, que es una estructura conteniendo un doble apuntador, el número de renglones (nRens) y el número de columnas (nCols). Usar este tipo de dato en todos los incisos i) e j).
i) Programar la función createMatrix y freeMatrix que crea (obtiene la memoria) y destruye (liberan memoria) matrices de dimensiones arbitrarias, respectivamente.
j) Hacer una función que calcula la transpuesta de una matriz de dimensiones arbitrarias, se le manda como parámetro la matriz original. La transpuesta se regresa en otra matriz.

k) Declarar el nuevo tipo de dato `Vector`, que es una estructura conteniendo un apuntador y el tamaño del vector . Usar este tipo de dato en todos los incisos siguientes.
l) Programar una función que calcula el producto de un escalar por un vector, el resultado es un vector. Más información en https://www.fisicapractica.com/producto-escalar-vector.php.
m) Programar una función que calcula el producto interno (o producto punto) de 2 vectores, el resultado es un escalar (se debe verificar que es posible operar los 2 vectores ya que tienen las mismas dimensiones). Más información en https://www.matetam.com/glosario/definicion/producto-interior-vectores.
n) Programar una función que calcula el producto matriz * vector, el resultado es un vector (la función debe de verificar que sus dimensiones permiten aplicar la operación). Más información en https://es.wikibooks.org/wiki/Álgebra_Lineal/Matriz_por_vector.