**Tarea 14:**

Fecha de entrega: ***ANTES*** *del 20 de Noviembre.*

Toda la tarea se debe de hacer usando memoria dinámica (`malloc()` y `free()` ) cuando sea requerida memoria.
A partir de ahora es obligatorio usar archivos .c y .h independientes para hacer las librerias que ocupan. El archivo `main.c` prácticamente solo contiene la funcion `int main(void){…}`

**A)** Hacer un programa que implementa el método de eliminación Gaussiana (calculando el sistema triangular reducido y usando finalmente Sustitución Hacia Atrás como se vio en CLASE) que vimos en clase, y cuyo algoritmo está explicado en las diapositivas Intro_SEL_1stSemDEMAT.pdf de la clase. La matriz **A** de coeficientes $a_{ij}$ y el vector **b** de términos independientes $b_i$ se leen desde archivos de texto. El vector $\boldsymbol{x}$ solución se escribe a un archivo de texto.

**B**) Hacer un programa que cree un archivo binario que tiene la siguiente estructura:

```
N
x1 y1 r1
x2 y2 r2
.  .  .
.  .  .
.  .  .
xN yN rN
```

Donde `N` (`int`) es un número de círculos, y cada i-ésimo círculo tiene su centro en la coordenada (`xi,yi`) del plano y radio `ri`. Los centros (`float`) está en el cuadradao en el 1er cuadrante definido por [0,10]x[0,10] y los radios (`float`) son aleatorios en el rango [0.1, 2].

Un ejemplo de 5 círculos sería:

```
5
   0.890033140823435   7.216268143326579
0.880613559580621
   1.545234272655767   0.976195387716772
0.839770708115929
   5.618885563009886   6.908503697481568
0.832907963429345
   4.658209789428525   9.884830199044989
0.808770870765704
   4.658209789428525  -0.115169800955011
0.808770870765704
```

**C*)*** Hacer un programa que lee el archivo binario del inciso ***A)***, lo guarda en un arreglo dinámico de tipo `Circulo` (una estructura) y, usando una función, calcula y muestra en pantalla el número de círculos que se traslapan entre sí a pares. Por ejemplo, en esta Figura

https://www.cimat.mx/~alram/elem_comp/circulos_no_intersectan.png

ningún circulo se traslapa con ningún otro, y en esta otra figura hay 3 traslapes a pares de círculos.

https://www.cimat.mx/~alram/elem_comp/circulos_si_intersectan.png

**D*)*** Dibujar en papel (y mandar una fotografía) el árbol de llamadas recursivas que se genera cuando mandar a calcular `fibonacciRecursive(7);` e indicar en que orden se hacen las llamadas (como lo hizo el profesor en clase). De acuerdo a tu figura ¿cuántas llamadas a la función se ejecutan? Corrobora ese número haciendo un printf en el código ejemplo fibo_recursive.c cada vez que se ejecuta la función.

***E***) Hacer una función (y probarla en `int main(){...}`) que de manera recursiva (no se usan ciclos `for, while`, etc.) calcula la

suma de los elementos de un vector de tamaño `n`.

**F**) Leer y entender (no reportar nada) el programa imprimeBinarioRecursive.c que imprime a pantalla la representación en binario de un `int`, **notar que el programa puede recibir argumentos desde consola (lo veremos en la clase del jueves 13-Nov)**.

**G**) Hacer una función (y probarla en `int main(){...}`) que de manera recursiva ordena un arreglo. Para ello, programar una función iterativa (una versión iterativa, al contrario de recursiva, significa que usa ciclos `for`, `while`, etc) que encuentra el elemento mas pequeño en un arreglo y lo intercambia por el elemento al inicio del arreglo. De esta manera la recursividad se define como: "*Si el arreglo tiene más de un elemento: para ordenarlo de manera recursiva hay que poner el elemento menor al inicio y ordenar recursivamente el resto del arreglo (de la segunda posición en adelante). De lo contrario, case base: un arreglo de un solo elemento ya está ordenado*".