Tarea 15 (Última Tarea 🙂):

Fecha de entrega: ***ANTES** del 30 de Noviembre.*

Toda la tarea se debe de hacer usando memoria dinámica (`malloc()` y `free()`) cuando sea requerida memoria (salvo que se especifique que se permite lo contrario).
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) Utilizar variables `static` para darle memoria a la función `int nextPrime()` que cada vez que se manda llamar regresa en orden ascendente el siguiente números primo menor a 1000 . Es decir, se comporta así:

```
int v
v = nextPrime(); //regresa 2 la primera vez que
se llama
v = nextPrime(); //regresa 3
v = nextPrime(); //regresa 5
...
v = nextPrime(); //regresa 997
v = nextPrime(); //regresa 2, se reinicia el
ciclo
```

B) Hacer una función `calculaXMinimo()` que encuentre el mínimo global (el valor de ***x*** del dominio que genera el valor más pequeño en la imagen) de una función matemática cualquiera ***f(x)*** en un intervalo ***a*** a ***b***. Para buscar en mínimo, ir avanzando ***x*** en pequeños incrementos desde ***a*** hasta ***b*** y evaluar ***f(x)*** para cada valor ***x***. La función `calculaXMinimo()` debe de servir para cualquier función

matematica que reciba un `double` como parámetro (valor de ***x***) y regrese un `double` por la izquierda (valor de ***f(x)***), para ello debe de recibir un apuntador a función. Probar que su programa sirve para 3 funciones diferentes (ejemplos: x^2, *sin(x)*, etc.)

C) Modificar el programa agentes.c visto en clase para que en lugar de imprimir textos en pantalla, cada agente haga una acción que se vea reflejada gráficamente en la consola. Para ello deben de agregar como propiedades de cada agente su coordenada (renglón,columna) en la que están situados en la pantalla.

D) Implementar la estructura de datos **`Pila`**. lo pueden hacer de manera estática (con un vector estático o dinámico) o de manera dinámica (con estructuras auto referenciadas y creando nodos), los métodos mínimos que se deben de programar son:

```
void iniPila(pila *);
int push(pila *, Item ); // regresa por la izquierda si pudo o no , por si se acaba la capacidad de la pila o la memoria
int pop(pila *, Item *); // regresa por la izquierda si pudo o no  (¿había elementos para sacar?), si no pudo Item tiene un elemento falso, no se debería de usar
```

E) Usando la pila del inciso anterior, programar una animación del problema de las Torres de Hanoi con solución recursiva que muestre en pantalla algo como lo que se muestra en https://www.mathsisfun.com/games/towerofhanoi.html

F) Leer (o releer como dice el autor) el excelente cuento La Biblioteca de Babel del excelentísimo Jorge Luis Borges: https://www.ingenieria.unam.mx/dcsyhfi/material_didactico/Literatura_Hispanoamericana_Contemporanea/Autores_B/BORGES/Babel.pdf

G) Dado el archivo `file_numbers_0_100.txt`, el cuál es un archivo de texto con `n` números aleatorios reales en `[0,100)`, con el formato:

```
n
num_1
num_2
num_3
.
.
num_n
```

(Abrirlo en su editor de texto para verlo y entender la estructura).

Hacer una función

```
int* leeImpares2Enteros(char *nombreArchivo, int *m);
```

que regresa por la izquierda un vector de tamaño `m <= n` con solo la parte entera de los `m` números cuya parte entera es impar. En la variable `m` regresa cuantos números se almacenaron en el vector. Probar que sirve la función en `int main(void){...}`

H) Modificar el programa `main_qsort.c` para que ordene un vector de `N` números complejos definidos en una `struct` con datos miembros `float real, imag;` El ordenado se hace en base a la magnitud del número complejo (la magnitud del número complejo se debe calcular usando una función que hace dicha tarea).

I) Modificar el codigo para que en lugar de que la lista quede desordenada, cada vez que se inserta un nodo, esta siempre sea una lista ordenada. Para ello modificar el código fuente en donde dice:

```
// AQUI HACER EL CAMBIO SOLICTADO
// PARA INSERTAR DONDE EL NODO LE TOCA
```

// PARA QUE LA LISTA ESTE' SIEMPRE ORDENADA

… y poner ahí en su lugar el código necesario que avanza en la la lista e inserta el nuevo nodo en la posicion que le corresponde.