Máximo de a 2. Proyecto final válido para el tercer corte. Para entregar en la plataforma Moodle el 6 de febrero máximo a las 12 pm. Obligatorio asistir el martes 04 de febrero al inicio de la hackathon, de no asistir el martes 04 de febrero su nota será 0.0 sin importar que haya subido el informe.

Cree un proyecto en Visual Studio, Replit o compile desde la consola de comandos. Utilice el lenguaje de programación C.

**Proyecto: Simulador de Batalla Espacial**

Descripción:

Desarrolla un juego de batalla espacial en consola de comandos que simule enfrentamientos entre naves espaciales. El juego debe ser dinámico, permitiendo a los usuarios ingresar comandos para controlar sus naves y atacar a las naves enemigas.

Características del Juego:

1. **Naves Espaciales:**
   * Cada nave espacial se representará como una estructura con atributos como nombre, tipo, salud, ataque, defensa, etc.
   * Deberás implementar la asignación dinámica de memoria utilizando apuntadores para gestionar la creación y destrucción de naves.
2. **Batallas:**
   * Las batallas se llevarán a cabo en turnos. Cada nave puede realizar acciones como atacar, defenderse o realizar maniobras evasivas.
   * Implementa funciones para calcular el daño causado por un ataque, considerando la defensa de la nave objetivo.
3. **Interfaz de Usuario:**
   * Diseña una interfaz de consola amigable que permita a los usuarios ingresar comandos para realizar acciones como atacar, defenderse, mostrar información, etc.

Requisitos Técnicos:

* Uso extensivo de punteros y asignación dinámica de memoria.
* Implementación de funciones para manipular y gestionar naves.
* Menús interactivos y mensajes claros para la interacción del usuario.

Ten en cuenta este esquema básico para comenzar. Te recomendaría desarrollar el proyecto de manera incremental, implementando y probando cada característica por separado. Aquí tienes una plantilla como base:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Estructura para la nave espacial

typedef struct {

char nombre[50];

int salud;

int ataque;

int defensa;

} NaveEspacial;

// Prototipos de funciones

NaveEspacial\* crearNave();

void destruirNave(NaveEspacial\* nave);

void guardarNaveEnArchivo(NaveEspacial\* nave, const char\* filename);

void cargarNaveDesdeArchivo(NaveEspacial\* nave, const char\* filename);

int main() {

NaveEspacial\* miNave = crearNave();

// Realizar acciones del juego, guardar y cargar desde archivos, etc.

destruirNave(miNave);

return 0;

}

NaveEspacial\* crearNave() {

NaveEspacial\* nuevaNave = (NaveEspacial\*)malloc(sizeof(NaveEspacial));

// Lógica para inicializar la nave

return nuevaNave;

}

void destruirNave(NaveEspacial\* nave) {

free(nave);

}

void guardarNaveEnArchivo(NaveEspacial\* nave, const char\* filename) {

FILE\* archivo = fopen(filename, "w");

if (archivo == NULL) {

fprintf(stderr, "Error al abrir el archivo para escritura.\n");

exit(EXIT\_FAILURE);

}

// Escribir los datos de la nave en el archivo

fclose(archivo);

}

void cargarNaveDesdeArchivo(NaveEspacial\* nave, const char\* filename) {

FILE\* archivo = fopen(filename, "r");

if (archivo == NULL) {

fprintf(stderr, "Error al abrir el archivo para lectura.\n");

exit(EXIT\_FAILURE);

}

// Leer los datos del archivo y cargarlos en la nave

fclose(archivo);

}

Solamente puede utilizar las librerías vistas en clase, el uso de otras librerías adicionales anulará el proyecto y este no se calificará.

**Entregue un informe que muestre el código, el pseudocódigo y el diagrama de flujo. Entregue pantallazos o video del debugging. El código debe poder copiarse y pegarse.**