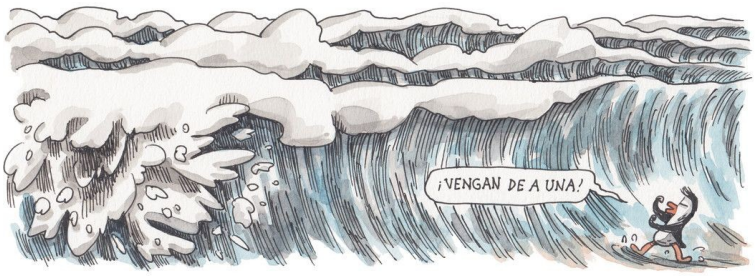



**Apellido, Nombre:**

.....

**Mail:**

.....

**Padrón:**

.....

| Teórico / Práctico - Entrego ..... hojas | | | | |
|---|---|---|---|---|
| 1: ..... | 2: ..... | 3: ..... | 4: ..... | Nota: ..... |

| RPL | | |
|---|---|---|
| 1: ..... | 2: ..... | Nota: ..... |

| Nota Final |
|---|
| ..... |

## Aclaraciones:

- **Antes de comenzar a resolver el parcial, complete sus datos en esta hoja, y al finalizarlo, firme todas las hojas.**
- **Los ejercicios deben ser implementados en el lenguaje de programación C, respetando las buenas prácticas de programación.**
- **Para cada ejercicio práctico se recomienda fuertemente realizar un análisis y un diagrama del problema y la solución.**
- **Se deben numerar TODAS las hojas e inicializarlas con nombre, apellido, padrón y cualquier otra información que considere necesaria.**
- **La aprobación del parcial está sujeta a la aprobación de los 2 módulos del mismo (teoría/práctica en papel y práctica en RPL).**

## Ejercicios:

1. **Dolores Umbridge** creó un algoritmo que determina si un alumno aprobó o no el examen. Dicho algoritmo devuelve **true** o **false** y se muestra a continuación.

```
bool aprobo_examen(respuesta_t respuestas[MAX_RESPUESTAS], int tope){
    int i = 0; int respuestas_bien = 0;
    while (i < tope){
      if (respuesta_correcta(respuestas[i])) respuestas_bien++;
    }
    return (respuestas_bien > MINIMO_RESPUESTAS_PARA_APROBAR);
}
```

**George** y **Fred** obtuvieron el código y se dieron cuenta que tiene una falla.

   a. ¿Cuál es la falla del algoritmo? Con el algoritmo así, ¿Qué debe cumplir el alumno para aprobar el examen?
   b. ¿Cuál es la diferencia entre enviar parámetros por valor y por referencia?

*Aclaración: Asumir que todas las funciones y constantes usadas funcionan y están correctamente definidas.*

2. **Severus Snape** tiene un armario con muchas pociones ya preparadas. Éstas están guardadas en frascos que pueden ser de 3 medidas diferentes. A su vez, cada una tiene distintos colores, intensidades, y efectos sobre quien la toma.
Los efectos, por convención, están representados por caracteres.
Las intensidades van de 1 a 5.
Para los colores no hay convención aún, pero es libre de crearla.
Cada poción debe tener un nombre y una lista de ingredientes (no más de 10 ingredientes por poción).
Cada ingrediente tiene nombre y cantidad.

   a. Crear un registro que sirva para almacenar una poción.
   b. ¿Cuántos bytes ocupa en memoria el registro para una arquitectura de 64 bits?.

*"Cada uno carga con su alma y con su cruz... Para dar batalla en las tormentas."* - El Umbral - Tabaré Cardozo.

3. **Gringotts** es el único banco de los magos, ubicado en el callejón *Diagon* y controlado por duendes. Fue fundado por el duende **Gringott**. Muchos magos y brujas británicos guardan ahi su dinero y cualquier objeto de valor en sus seguras bóvedas subterráneas.

   En dicho banco se está efectuando un reordenamiento de las bóvedas, ya que aquellas con mayor fortuna deben estar en lo más profundo del banco y las de menos fortuna, más cerca de la superficie.

   ```
   typedef struct boveda{
         char duenio[MAX_DUENIO];
         int fortuna;
   } boveda_t;
   ```

   a. Cree un procedimiento que, dado un vector de bóvedas, lo ordene de menor a mayor según su fortuna.
   b. Explique la búsqueda binaria. No debe escribir el algoritmo, solo explicarla.

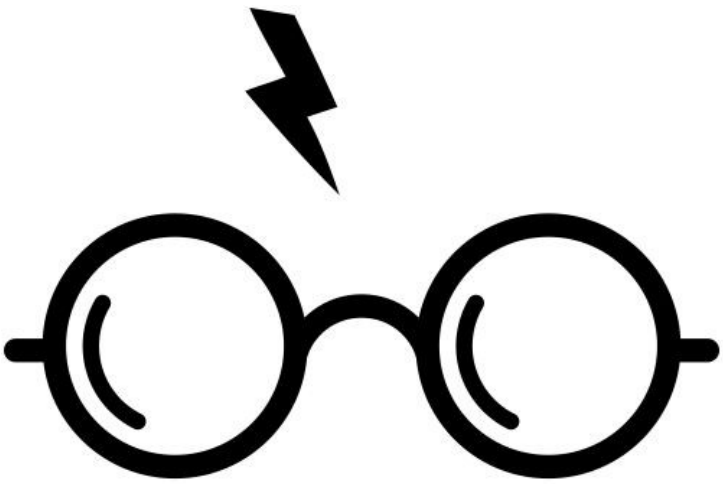
4. **Griphook** decidió abrir su propio banco, varios de los clientes de **Gringotts** decidieron probar suerte en el emprendimiento de **Griphook** por lo que sus bóvedas debieron ser trasladadas al nuevo edificio.

   Debido a esto, **Gringotts** debe hacer una actualización de sus bóvedas, eliminando aquellas que decidieron cambiar de banco.

   Se tienen 2 vectores de bóvedas, **ordenados por dueño**, el primero contiene las bóvedas de **Gringotts** (antes de la mudanza) y el segundo las bóvedas que deben eliminarse.

   a. Realizar un procedimiento que cree un tercer vector con las bóvedas que le quedan a **Gringotts**. Adicionalmente el procedimiento debe imprimir por pantalla que porcentaje de la fortuna han perdido.

   b. Si **Griphook** debe eliminar una bóveda de su vector de bóvedas, sabiendo que el vector está ordenado por dueño y tiene el nombre del dueño a eliminar, ¿cómo lo haría sin utilizar un vector auxiliar?. Recuerde que el vector debe quedar ordenado y sin huecos. No debe escribir el algoritmo, solo explicarlo.



*"Cada uno carga con su alma y con su cruz... Para dar batalla en las tormentas."* - El Umbral - Tabaré Cardozo.

## RPL:

1. Hagrid tiene una huerta de calabazas, quiere saber cuanto dulce y cuanta sobra para su lumbricario puede sacar de ella.
   Cada calabaza tiene,
   Un estado:
   - Verde, no puede usarse para dulce.
   - Madura, puede usarse entera para dulce.
   - Pasada, solo puede usarse la mitad de la calabaza para dulce y la otra mitad para el lumbricario

   Un peso (en kilos):
   - De ese peso, la mitad es cáscara y semillas, que no pueden ser usados para dulce pero si para el lumbricario.

   Sabiendo que solo cosechará las maduras y pasadas, crear una función que devuelva la cantidad de calabazas verdes que quedan sin cosechar (como retorno de ella), la cantidad (en kilos) de calabaza para dulce (en un parámetro pasado por referencia) y la cantidad (en kilos) de calabaza para el lumbricario.

   *Aclaración: Todas las variables correspondientes a cantidades deben ser de tipo **int**.*

```
#define MAX_CALABAZAS 400
typedef struct calabaza{
    char estado; // V: verde - M: madura - P: pasada
    int peso;
} calabaza_t;
int hacer_dulce(calabaza_t calabazas[MAX_CALABAZAS], int tope_calabazas, int*
kilos_dulce, int* kilos_lumbricario){
    // Aqui va tu codigo...
}
```

2. *Harry* está caminando por *Hogwarts* a altas horas de la noche, cosa que está prohibido en la escuela. Los profesores escucharon ruidos extraños por los pasillos y están buscándolo.
   Por suerte, trae consigo el *Mapa del Merodeador* y puede saber que donde está cada uno de los profesores. El mapa puede representarse como una matriz, *Harry* está en la posición (0,0) y es representado por una **H**.
   En él vio que los que lo buscan son:
   - *Severus Snape* representado por una **S**.
   - *Remus Lupin* representado por una **L**.
   - *Alastor Moody* representado por una **M**.
   - *Dolores Umbridge* representada por una **U**.

   Si bien puede verlos, necesita saber cual está mas cerca para saber con quien tendrá que lidiar primero.
   Se pide realizar una función que devuelva el caracter que representa al profesor que se encuentra mas cerca.
   Como *Harry* está en la posición (0,0) de la matriz, la distancia puede calcularse como la suma de las coordenadas de la posición del profesor.
   En caso de haber 2 o más profesores a la misma distancia, devolver **X**.

```
#define MAX_MAPA 400
char profesor_mas_cercano(char mapa[MAX_MAPA][MAX_MAPA], int alto, int ancho){
    // Aqui va tu codigo...
}
```

*"Cada uno carga con su alma y con su cruz... Para dar batalla en las tormentas."* - El Umbral - Tabaré Cardozo.

////////////////////////////////////////////////////////////////////////TEORICO-PRACTICO////////////////////////////////////////////////////////////////////////

1a) La falla es que la variable "i" que se usa para avanzar sobre el vector de respuestas nunca cambia de valor.

Con el algoritmo así, ningún alumno aprobaria el examen ya que al no cambiar el valor de "i", nunca superaría el valor del tope y la funcion "while" nunca cortaría
(bucle infinito).

Se podría utilizar un "for" en vez de un "while" por saber de antemano la cantidad de iteraciones, es decir el tope.

1b) La diferencia entre enviar parámetros por valor y por referencia, es que por referencia cada modificación que se le haga a un parámetro dentro de una determinada
función, también es plasmada en la función de la cual se llamo. En cambio, esto no sucede cuando se envia un parámetro por valor, la modificación que se le haga a
dicho parámetro solo se refleja dentro de la función que lo recibió.

2a)

```
#define MAX_LETRAS 20
#define MAX_INGREDIENTES 10

typedef struct ingrediente{
  char nombre[MAX_LETRAS];
  int cantidad;
} ingrediente_t;

typedef struct pocion{
  char nombre[MAX_LETRAS];
  ingrediente_t ingredientes[MAX_INGREDIENTES];
  int medida; // 1: pequenio - 2: mediano - 3: grande
  char color[MAX_LETRAS];
  int intensidad; // de 1 a 5
  char efecto;
} pocion_t;
```

2b) Primero debo calcular cuanto ocupa cada tipo de dato dentro del struct en una arquitectura de 64 bits:

char nombre[MAX_LETRAS]; = 1 byte * 20 (MAX_LETRAS) = 20 bytes
int cantidad; = 4 bytes

ej: ingrediente_t sangre_pato = 24 bytes

char nombre[MAX_LETRAS]; = 1 byte * 20 (MAX_LETRAS) = 20 bytes
ingrediente_t ingredientes[MAX_INGREDIENTES]; = 24 bytes * 10 (MAX_INGREDIENTES) = 240 bytes
int medida; = 4 bytes
char color[MAX_LETRAS]; = 1 byte * 20 (MAX_LETRAS) = 20 bytes
int intensidad; = 4 bytes
char efecto; = 1 bytes

ej: pocion_t multijugos = 289 bytes

3a) Análisis:

Debemos crear un procedimiento que reciba un vector de bóvedas de Gringott con su respectivo tope, que ordene de menor a mayor las bóvedas según su fortuna.

Podemos usar un método de ordenamiento como bubble sort, utilizando un doble ciclo for. Se compara la fortuna de la bóveda actual con la de la bóveda siguiente
y en caso de que sea mayor, se intercambian las bóvedas. De esta manera la bóveda con mayor fortuna se ubicaría al final del vector y se le resta un valor al tope del

segundo ciclo, ya que sabremos que la última posición va a estar ocupada por la mayor riqueza y no va a ser necesario comparar. Luego se repite nuevamente este
procedimiento hasta que el vector quede ordenado de menor a mayor fortuna.

Código:

```
typedef struct boveda{
  char duenio[MAX_DUENIO];
  int fortuna;
} boveda_t;

void ordenar_bovedas(boveda_t bovedas[MAX_BOVEDAS], int tope_bovedas){
  for(int i = 0; i < tope_bovedas; i++)
    for(int j = 0; j < (tope_bovedas - i); j++)
      if(bovedas[j].fortuna > bovedas[j+1].fortuna)
        intercambiar_bovedas(&(boveda[j]), &(boveda[j+1]) );
}

void intercambiar_bovedas(boveda_t* boveda_uno; boveda_t* boveda_dos){
  boveda_t aux;
  aux = boveda_uno;
  boveda_uno = boveda_dos;
  boveda_dos = aux;
}
```

3b) La búsqueda binaria puede ser utilizada únicamente cuando el vector en el cual se realiza la operación posee sus elementos ordenados. Este método consiste en buscar
el valor que se encuentra en el medio del vector, comparar con el valor buscado, y trabajar con la mitad de vector correspondiente (La primera mitad si el valor buscado es menor al valor
del medio o la segunda mitad si es mayor). Asi repetir hasta llegar a la posición del valor buscado o bien determinar que no se encuentra cuando los valores inicio y fin se cruzan.

4a) Análisis:

Debemos hacer un procedimiento que reciba por parámetros dos vectores de bóvedas ordenados por dueño, el primero contiene las bóvedas de Gringotts (antes de
la mudanza) y el segundo las bóvedas que deben eliminarse, y que cree un tercer vector con las bóvedas que le quedan a Gringotts. Además, este debe imprimir por
pantalla que porcentaje de la fortuna ha perdido.

Primero, debemos definir un nuevo vector de tipo de dato boveda_t que va a ser el vector final que contenga las bóvedas que quedan en el banco de Gringotts,
un tope para este vector inicilizado en cero, dos variables de tipo número de coma flotante ambas inicializadas en cero, una para contar la fortuna total y otra para
la perdida.

Se recorren los vectores recibidos, para cada posición del vector de Gringotts antes de la mudanza, se debe ir sumando la riqueza de todas las bóvedas a la
variable que definimos al inicio de riqueza total. Además, se van comparando los strings que contienen el nombre de los dueños y en caso de haber coincidencia, se
aumenta el tope del tercer y nuevo vector, se copia los datos de la bóveda en este último mencionado y se suma la riqueza de esta en la varible de riqueza perdida.

Finalmente, una vez recorridos los vectores, se calcula el porcentaje de fortuna perdida por Gringotts y se imprime por pantalla.

Código:

*"Cada uno carga con su alma y con su cruz... Para dar batalla en las tormentas."* - El Umbral - Tabaré Cardozo.

```
#define COINCIDENCIA 0

void transferir_bovedas(boveda_t gringotts[MAX_BOVEDAS], int tope_gringotts,
boveda_t griphook[MAX_BOVEDAS], int tope_griphook){
  boveda_t gringotts_final[MAX_BOVEDAS];
  float fortuna_perdida = 0;
  float porcentaje_perdido;
  int comparar;
  float fortuna_total = 0;
  int bovedas_trasladadas = 0;
  int i = 0;

  while(i <= tope_griphook){
    for(int j = 0; j < tope_gringotts; j++){
      fortuna_total += gringotts[j].fortuna;

      comparar = strcmp(gringotts[j].duenio, griphook[i].duenio);

      if(comparar == COINCIDENCIA){
        fortuna_perdida += gringotts[j].fortuna;
        i++;
      }
      else{
        copiar_boveda(&(gringotts[j]), &(gringotts_final[bovedas_trasladadas])
);
        bovedas_trasladadas++;
      }
    }
    i++;
  }
  porcentaje_perdido = (fortuna_perdida * 100) / fortuna_total;

  printf("Porcentaje de riqueza perdida: "%i" %", porcentaje_perdido);
}

void copiar_boveda(boveda_t* boveda_uno; boveda_t* boveda_dos){
  boveda_uno = boveda_dos;
}
```

4b) Recorrería el vector de la boveda hasta encontrar la posición de la bóveda a eliminar, y a partir de esa posición copiaría la información de la siguiente boveda en
la posición de la bóveda actual, así hasta recorrer todo el vector. Finalmente restaría uno al tope de dicho vector bóveda.

////////////////////////////////////////////////////////////////////////////////
RPL//////////////////////////////////////////////////////////////////////////////
1) Análisis:
Debemos crear una función que devuelva la cantidad de calabazas verdes que quedan sin cosechar (como retorno de ella), la cantidad (en kilos) de calabaza
para dulce (en un parámetro pasado por referencia) y la cantidad (en kilos) de calabaza para el lumbricario. Por la firma de la función, sabemos que la huerta de
Hagrid es representada por un vector.
Primero debemos declarar una variable de tipo entero inicializada en cero para contar las calabazas que queden sin cosechar, y luego hay que recorrer el
vector y verificar el estado de las calabazas:
-Si está verde: Se aumenta el contador de calabazas sin cosechar.
-Si está madura: La mitad de su peso se le suma a la variable de kilos para dulce recibida por referencia.
La otra mitad (cáscara y semillas) se le suman a la variable de kilos para lumbricario también recibida por referencia.

-Si está pasada: La mitad de la mitad de su peso, es decir un cuarto, se le suma a la variable de kilos para dulce recibida por referencia. El resto, tres cuartos de su peso, (cáscara, semillas y partes pasadas) se le suman a la variable de kilos para lumbricario también recibida por referencia.

Una vez finalizado el recorrido del vector, se devuelve como retorno de la función el valor de calabazas maduras en el contador de calabazas sin cosechar.

Código:

```
#define MADURA 'M'
#define PASADA 'P'

typedef struct calabaza{
  char estado; // V: verde - M: madura - P: pasada
  int peso;
} calabaza_t;

int hacer_dulce(calabaza_t calabazas[MAX_CALABAZAS], int tope_calabazas, int*
kilos_dulce, int* kilos_lubricario]){
  int sin_cosechar = 0;

  for(int i = 0; i < tope_calabazas; i++){
    if(calabazas[i].estado == MADURA){
      *kilos_dulce += calabazas[i].peso / 2;
      *kilos_lubricario += calabazas[i].peso / 2;
    }

    else if(calabazas[i].estado == PASADA){
      *kilos_dulce += calabazas[i].peso / 4;
      *kilos_lubricario += calabazas[i].peso / 0.75;
    }
    else{
      sin_cosechar++;
    }
  }
  return sin_cosechar;
}
```

2) Análisis:

Se nos pide realizar una función que devuelva el caracter que representa al profesor que se encuentra más cerca de Harry. La distancia puede calcularse como la suma de las coordenadas de la posición del profesor, sabiendo que Harry está en la posición (0,0) y que el mapa de la escuela está representado por una matriz cuadrada.

Primero debemos definir una variable de tipo entero para la distancia, inicializada con el valor maximo de distancia posible, y otra de tipo caracter para almacenar la inicial del profesor que se encontrará más cerca. Luego recorremos la matriz, y en caso de encontrarnos con la posición de un profesor, se calcula la distancia del mismo con respecto a Harry. Se compara la distancia del encontrado con la anterior (si es el primero, con la max distancia inicializada al comienzo), y en caso de encontrarse más cerca, se guarda su inicial en la variable de tipo char. Si se encuentran a la misma distancia, es un "empate" y se guarda el caracter 'X'.

Una vez finalizado el recorrido de la matriz que representa el mapa, vamos a obtener en la variable de tipo char la inicial del profesor que se encuentra más cerca de Harry (o 'X' si son más de uno), y esta se devuelve por retorno de la función.

Código:

*"Cada uno carga con su alma y con su cruz... Para dar batalla en las tormentas."* - El Umbral - Tabaré Cardozo.

```
#define MAX_MAPA 400
#define MAX_DISTANCIA 800
#define SNAPE 'S'
#define LUPIN 'L'
#define MOODY 'M'
#define DOLORES_UMBRIDGE 'U'
#define VARIOS_CERCA 'X'

char profesor_mas_cercano(char mapa[MAX_MAPA][MAX_MAPA], int alto, int ancho){
  int distancia;
  int menor_distancia = MAX_DISTANCIA;
  char mas_cerca;

  for(int i = 0; i < alto; i++){
    for(int j = 0; j < ancho; j++){

      if( (mapa[i][j] == SNAPE) || (mapa[i][j] == LUPIN) || (mapa[i][j] ==
MOODY) || (mapa[i][j] == DOLORES_UMBRIDGE) ){
        distancia = i + j;

        if(distancia < menor_distancia){
          menor_distancia = distancia;

          mas_cerca = mapa[i][j];
        }
        else if(distancia == menor_distancia){
          mas_cerca = VARIOS_CERCA;
        }
      }
    }
  }
  return mas_cerca;
}
```