Nombre y Apellido: ..... N° Legajo: .....

***Primer Parcial de Programación Imperativa***

28/04/2023

| | *Ejercicio 1* | *Ejercicio 2* | *Ejercicio 3* | *Nota* |
|---|---|---|---|---|
| **Calificación** | **/3** | **/3.5** | **/3.5** | |

- **Condición mínima de aprobación: Sumar 5 (cinco) puntos.**
- **Los ejercicios que no se ajusten estrictamente al enunciado, no serán aceptados.**
- **No usar variables globales ni static.**
- **No es necesario escribir los #include**
- **Escribir en esta hoja Nombre, Apellido y Legajo**

## *Ejercicio 1*

Escribir la función **elimina** que recibe tres strings `s1`, `s2` y `s3` y **elimina** de `s1` aquellos caracteres que están presentes en s2 o en s3 **en la misma posición** que en s1.

Ejemplo de uso:

```
int main(void) {
   char s[] = "abc";
   elimina(s, "123", "cab");
   assert(strcmp(s, "abc") == 0); // No se eliminan caracteres

   elimina(s, "axc", "xbc");
   // Se elimina la a porque está en s2 en la misma posición
   // se elimina la b porque está en s3 en la misma posición
   // Se elimina la c porque está en s2 o en s3 en la misma posición
   assert(strcmp(s, "") == 0);,

   char t[] = "abc 123";
   elimina(t, "b", "1");
   assert(strcmp(t, "abc 123") == 0);  // No se eliminan caracteres

   elimina(t, "aaaaaaaaaaaaaaaaaaaaa", "2222222222222222");
   assert(strcmp(t, "bc 13") == 0);

   elimina(t, "", "");  // No se eliminan caracteres
   assert(strcmp(t, "bc 13") == 0);

   puts("OK!");
   return 0;
}
```

## *Ejercicio 2*

Escribir una función **`verifica`** que recibe como único parámetro una "matriz" de enteros de `NxN`, donde `N` es una constante previamente definida, de tipo entero y múltiplo de 3, por ejemplo 3, 6, 9, etc.

La función debe **retornar** 1 si se cumplen **todas** las siguientes condiciones:

- Todos los elementos de la matriz están entre 1 y 3*N inclusive
- Cada submatriz de 3x3 debe tener elementos sin repetir
- Al sumar los elementos de cada submatriz de 3x3 se obtiene el mismo resultado

Las submatrices son similares a las del juego de Sudoku, comienzan en las posiciones (0,0), (0,3), (3,0), (3,3), (0,6), etc.

Ejemplos:

Si N es 3 y recibe la siguiente matriz, debe retornar 1:

| | | |
|---|---|---|
| 1 | 3 | 2 |
| 9 | 8 | 6 |
| 4 | 7 | 5 |

Si N es 6 y recibe la siguiente matriz, debe retornar 1, ya que no hay repetidos en cada una de las cuatro submatrices y todas suman lo mismo

| | | | | | |
|---|---|---|---|---|---|
| 10 | 3 | 2 | 3 | 8 | 7 |
| 12 | 8 | 6 | 6 | 2 | 4 |
| 4 | 7 | 5 | 5 | 12 | 10 |
| 2 | 12 | 6 | 10 | 4 | 5 |
| 5 | 10 | 3 | 7 | 2 | 9 |
| 8 | 4 | 7 | 3 | 6 | 11 |

Si N es 3 y recibe la siguiente matriz, debe retornar 0 (el 15 está fuera del rango 1 a 3*N)

| | | |
|---|---|---|
| 1 | 3 | 2 |
| 9 | 8 | 6 |
| 4 | 7 | ***15*** |

Si N es 6 y recibe la siguiente matriz, debe retornar 0. Si bien las 4 submatrices suman lo mismo, en una se repiten valores:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 2 | ***3*** | 8 | ***5*** |
| 9 | 8 | 6 | 6 | 2 | 4 |
| 4 | 7 | 5 | ***5*** | 9 | ***3*** |
| 2 | 9 | 6 | 1 | 4 | 5 |
| 5 | 1 | 3 | 7 | 2 | 8 |
| 8 | 4 | 7 | 3 | 6 | 9 |

## *Ejercicio 3*

Implementar una función **`wordle`** que resuelva un tablero del juego Wordle (también conocido como palabra del día). La función recibe:

- Una **palabra secreta** de `COLS` caracteres (no es necesario validarlo)
- Una **matriz de caracteres** de tamaño `FILS x COLS` donde cada fila es un intento del usuario para resolver la palabra secreta
- La **cantidad de intentos** del usuario (un entero menor o igual a `FILS`)

La función debe dejar en **otra matriz** una marca `V`, `A` o `G` para cada letra de cada intento del usuario donde:

- **`V: VERDE`** significa que la letra está en la palabra y en la posición CORRECTA.
- **`A: AMARILLO`** significa que la letra está presente en la palabra pero en la posición INCORRECTA.
- **`G: GRIS`** significa que la letra NO está presente en la palabra.

hasta que se marque el primer intento correcto (todas las letras verdes) o se hayan analizado todos los intentos.

La función debe **retornar** un entero con el número del primer intento correcto (todas las letras verdes) o -1 si ningún intento es correcto.

Notar que tanto la palabra oculta como las palabras de los intentos pueden contener letras repetidas. En ese caso, las pistas son independientes para cada letra y tienen prioridad: verde tiene mayor prioridad al amarillo.

Ejemplos:

Con `FILS = 6` y `COLS = 5`, donde la palabra secreta a adivinar es **`R O S A S`** y el usuario hace los siguientes 6 intentos:

```
F A R O L
S A C O S
R A T O S
R O S A S
C O S A S
T E C L A
```

la función retorna 4 porque el cuarto intento es correcto y se obtiene:

```
G A A A G
A A G A V
V A G A V
V V V V V
? ? ? ? ?
? ? ? ? ?
```

- Para **`F A R O L`** se marcan las letras A, R y O que están en la palabra pero en la posición incorrecta y queda `G A A A G`
- Para `S A C O S` se marca la primera S en amarillo y la segunda S en verde y queda `A A G A V`
- Para `R A T O S` se marca en verde la S pues está en la posición correcta y no se avisa si hay una letra repetida y queda `V A G A V`
- Para `R O S A S` como la palabra coincide con todas las letras se marcan como verdes y queda `V V V V V`
- Para el quinto y sexto intento no se marca nada porque el cuarto fue correcto