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

Primer Parcial de Programación Imperativa

23/09/2022

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

- **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 una función que reciba una matriz cuadrada de dimensión N (N es una constante previamente definida) y retorne 1 si la misma cumple con las siguientes condiciones:

1) Contiene todos los números de **1 a N * N inclusive**

2) En cada fila contiene **números consecutivos**, no necesariamente ordenados

Ejemplos:

Si N es 5, se debe cumplir que tenga todos los números del 1 al 25 y la primer fila tenga los números del 1 al 5 (no necesariamente en ese orden), la segunda fila los números del 6 al 10, la tercera fila del 11 a 15, la cuarta fila del 16 al 20 y la última del 21 al 25.

Si N es 3, se debe cumplir que tenga los números de 1 a 9 y que la primer fila tenga los números del 1 al 3, la segunda del 4 al 6 y la última del 7 al 9.

Si N es 4 y recibe la siguiente matriz, debe retornar 1:

3	4	1	2
8	7	6	5
9	10	11	12
15	16	13	14

Si N es 4 y recibe la siguiente matriz, debe retornar 0 (repite el 1 en la primer fila y no está el 4):

3	***1***	***1***	2
8	7	6	5
9	10	11	12
15	16	13	14

Si N es 4 y recibe la siguiente matriz, debe retornar 0 (está el 11 en la última fila, y no está el 14):

3	1	4	2
8	7	6	5
9	10	11	12
15	16	13	***11***

Ejercicio 2

Escribir una función **`palEnMat`** que reciba:

- una **matriz** cuadrada de chars de dimensión N (con N constante simbólica previamente definida)
- un entero que indica una **fila**
- un entero que indica una **columna**
- un **string**

y retorne 1 **si el string se encuentra dentro de la matriz**, ya sea en sentido horizontal, vertical o diagonal y en cualquier dirección, donde el primer carácter del **string** se encuentra en la **fila** y **columna** indicada.

Ejemplos:

Para la siguiente matriz y con N = 6

L	O	B	R	A	Z
F	H	R	G	O	R
C	O	S	A	D	A
R	L	A	H	N	N
W	A	O	Y	U	T
Q	S	G	S	M	A

retornaría 1 si recibe:

- fila = 5, columna = 1, string = "SAL"
- fila = 2, columna = 0, string = “COSA”
- fila = 2, columna = 0, string = “CHB”
- fila = 2, columna = 0, string = “CLOS”
- fila = 2, columna = 0, string = ""
- fila = 2, columna = 0, string = "C"

retornaría 0 si recibe:

- fila = 5, columna = 1, string = "sal"
- fila = 5, columna = 1, string = "SALA"
- fila = 2, columna = 0, string = “COSADAS”
- fila = 6, columna y string no importa, la fila está fuera de rango

Ejercicio 3

Se tiene en un vector los factores que se obtienen como resultado de la factorización de un número. Los factores pueden estar repetidos, y están ordenados de menor a mayor. El valor -1 indica el final del vector.

Por ejemplo, el factoreo de 12 es 2 * 2 * 3, entonces el vector es {2, 2, 3, -1}.

Escribir la función **`factoreo`** que reciba el vector con los factores y obtenga el número original y deje en el vector sólo los valores diferentes.

Nota: el vector de factores es válido, no contiene ceros ni negativos, salvo por la marca de final (-1).

Ejemplo de Invocación:

```
int main(void) {
     int factores[] = {2, 2, 2, 3, 3, 4, 5, 5, 6, 6, 6, 6, 7, -1};
     long resultado = factoreo(factores);
     assert(resultado == 65318400);
     assert(factores[0] == 2);
     assert(factores[1] == 3);
     assert(factores[2] == 4);
     assert(factores[3] == 5);
     assert(factores[4] == 6);
     assert(factores[5] == 7);
     assert(factores[6] == -1);
     int factores2[] = {2, -1};
     resultado = factoreo(factores2);
     assert(resultado == 2);

     assert(factores2[0] == 2);
     assert(factores2[1] == -1);
     int factores3[] = {11, 11, 13, -1};
     resultado = factoreo(factores3);
     assert(resultado == 1573);  // 11 * 11 * 13
     assert(factores3[0] == 11);
     assert(factores3[1] == 13);
     assert(factores3[2] == -1);
     int factores4[] = {-1};
     resultado = factoreo(factores4);
     assert(resultado == 0);
     assert(factores4[0] == -1);
     int factores5[] = {1, -1};
     resultado = factoreo(factores5);
     assert(resultado == 1);
     assert(factores5[0] == 1);
     assert(factores5[1] == -1);
     puts("OK!");
     return 0;
}
```