

Segundo Parcial de Laboratorio

Algoritmos y Estructura de Datos II

TEMA B

Ejercicio 2

Implementar el TAD *Dominó* que representa una ficha del juego denominado Dominó. No es necesario conocer el juego, solo se deben seguir ciertas pautas que se detallan a continuación.

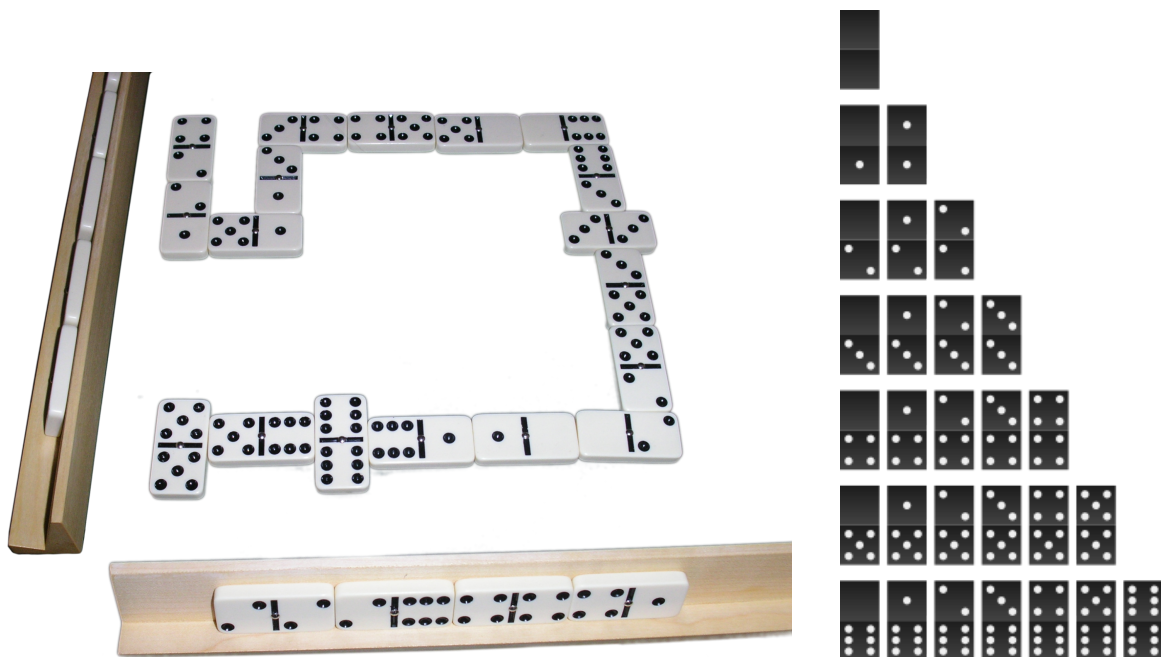


Imagen ilustrativa de una partida de dominó

El juego Dominó cuenta con 28 fichas rectangulares donde en cada una de ellas figuran dos números. Para referirnos a los números vamos a pensar que están orientadas verticalmente:



Entonces, **(a)** es la ficha con un **dos** arriba y un **cuatro** abajo y **(b)** la ficha con **cero** arriba y un **cinco** abajo. Los números pueden ser 0, 1, 2, 3, 4, 5 o 6. Para referirnos a una ficha usaremos la notación $n:m$. Entonces la ficha (a) se puede escribir como $2:4$ y la ficha (b) $0:5$.

El TAD tiene la siguiente interfaz:

Función	Descripción
<code>domino domino_new(int num_up, int num_down)</code>	Crea una ficha con numeración <code>num_up</code> arriba y con numeración <code>num_down</code> abajo
<code>bool domino_is_double(domino p)</code>	Indica si la ficha es de la forma <code>n:n</code>
<code>bool domino_eq(domino p1, domino p2)</code>	Indica si la ficha <code>p1</code> es la misma ficha que <code>p2</code>
<code>bool domino_matches(domino p_top, domino p_bottom)</code>	Indica si la ficha <code>p_top</code> encaja con la ficha <code>p_bottom</code> ubicando a la primera arriba y la segunda por debajo.
<code>domino domino_flip(domino p)</code>	Da vuelta una ficha, haciendo que la numeración que tenía por debajo ahora esté arriba y viceversa. No genera una nueva instancia sino que modifica a <code>p</code> .
<code>void domino_dump(domino p)</code>	Muestra una ficha por pantalla
<code>domino domino_destroy(domino p)</code>	Destruye una instancia del TAD <i>Domino</i> , liberando toda la memoria utilizada

En `domino.c` deben definir la estructura de representación interna `struct _s_domino` así como la invariante de representación que **se debe verificar cuando corresponda**.

Equivalencia entre fichas

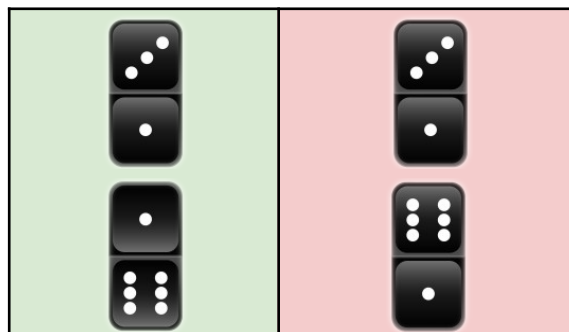
Si una ficha `p1` tiene los mismos números que una ficha `p2`, **sin importar** arriba o abajo, se considera que `p1` y `p2` **son la misma ficha**.



Por lo tanto en ese caso `domino_eq(p1, p2)` debe ser igual a `true`.

Encaje de fichas

Se considera que una ficha `p_top` **encaja** con una ficha `p_bottom` si al colocar `p_top` por encima de `p_bottom` los números que se tocan coinciden, en caso contrario no encajan.



En otras palabras, el número de abajo de `p_top` debe coincidir con el número de arriba de `p_bottom`. Además está prohibido intentar encajar una ficha con sígla misma, por este motivo **se pide como precondition** de la función `domino_matches()` que **no sea cierto** `domino_eq(p_top, p_bottom)`. Notar que **3:1 encaja con 1:6 pero no con 6:1**.




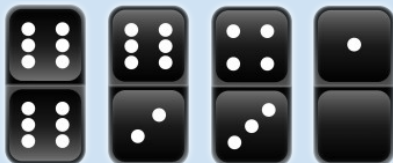


a) Implementar en `domino.c` el TAD *Dominó* siguiendo las pautas de encapsulamiento vistas en la materia.

b) Completar la función `chain_dominoes()` en el archivo `domino_helpers.c`:

```
unsigned int chain_dominoes(domino pieces[], unsigned int size)
```

que dado un arreglo de fichas de dominó `pieces[]` de tamaño `size` intenta encadenar todas las fichas encajándolas. La función devuelve la cantidad de fichas que pudo hacer coincidir. El arreglo debe tener al menos una ficha (`size > 0`), y el primer elemento de `pieces[]` debe ser un dominó doble (chequear eso como pre condición). Si una ficha p_n no encaja con la siguiente ficha p_{n+1} se debe invertir a p_{n+1} . Si aún así no encajan, entonces la cadena termina en ese punto. Un par de ejemplos:

```
matches = chain_dominoes(pieces, size);
```

Antes	Después
 <pre>pieces == [4:4, 3:4, 3:5, 1:5] size == 4</pre>	 <pre>pieces == [4:4, 4:3, 3:5, 5:1] matches == 4</pre>
 <pre>pieces == [6:6, 2:6, 3:4, 1:0] size == 4</pre>	 <pre>pieces == [6:6, 6:2, 4:3, 1:0] matches == 2</pre>
 <pre>pieces == [3:3, 2:4, 5:6] size == 3</pre>	 <pre>pieces == [3:3, 4:2, 5:6] matches == 1</pre>

IMPORTANTE: El TAD *Dominó* y el programa en `main.c` deben estar libres de *memory leaks* y de *invalid reads / writes*

TIP: Pueden probar el TAD Dominó usando el programa de `main.c` dejando sin completar la implementación de `chain_dominoes()`. Van a obtener como salida todas las fichas de dominó ingresadas.