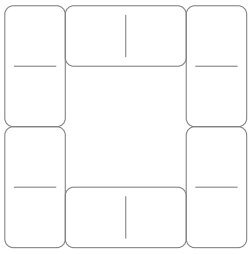
Problema B
Dominó

Un juego de dominó tradicional consiste en un conjunto de fichas rectangulares de 2×1, donde cada mitad contiene un número entre 0 y 6. Cada combinación de pares de números aparece exactamente una vez, pero sin considerar el orden. Por ejemplo, la ficha 1–6 es la misma que la 6–1. Esto da un total de 28 fichas para el dominó tradicional.

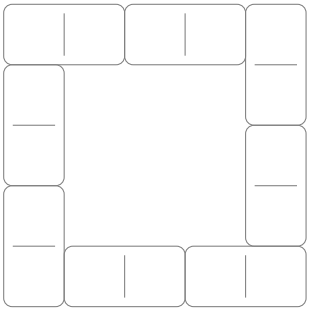
Dado un entero n mayor que cero, el juego de dominó puede ser generalizado para que cada pieza contenga números entre 0 y $n-1$. En este dominó generalizado cada combinación de pares de números también aparece exactamente una vez. Por ejemplo para $n = 2$ las fichas son 0–0, 0–1 y 1–1. Llamaremos un n-dominó al juego donde cada ficha contiene números entre 0 y $n-1$, y cada combinación de pares de números aparece exactamente una vez. Por ejemplo, el dominó tradicional es un 7-dominó. Por otro lado, el conjunto (0–0, 0–2, 2–2) está formado por fichas con valores entre 0 y 2, pero no conforman un 3-dominó, pues hay fichas que faltan.

Un n-dominó es tan versátil como una baraja de cartas, en cuanto a que ambos permiten jugar una gran variedad de juegos. Uno de estos es el del cuadrado. En el juego del cuadrado se te entrega un n-dominó y un valor k. El objetivo del juego es construir un cuadrado usando todas las fichas que conforman el n-dominó, de forma que para cada lado la suma de todos los valores en las fichas sea k. A un cuadrado construido con todas las fichas de un n-dominó en que todos los lados sumen k lo llamaremos un cuadrado (n, k).

Para construir un cuadrado válido la cantidad de fichas verticales y horizontales debe ser la misma. A continuación se muestra una imagen de un cuadrado inválido y uno válido. El cuadrado de la izquierda es inválido pues los lados horizontales están formados por una ficha completa y dos mitades, mientras que los verticales por dos fichas completas. El cuadrado de la derecha es válido y cada lado, horizontal o vertical, está formado por dos fichas completas y una mitad. Notar que todas las fichas están contenidas en un solo lado salvo por las fichas de las esquinas que pertenecen a dos lados simultáneamente.



Cuadrado inválido

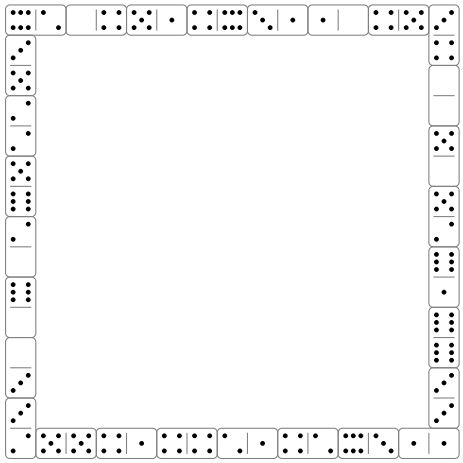


Cuadrado válido

Para un valor n puede ser el caso que dada la cantidad de fichas que conforman un n-dominó sea imposible construir un cuadrado válido. Por ejemplo, el 2-dominó está formado por 3 fichas (0–0, 0–1 y 1–1) y no es posible construir un cuadrado válido con ellas. De hecho el primer n-dominó con el cuál es posible construir un cuadrado válido es el 7-dominó. Por otro lado, para un n-dominó donde sí es

posible construir un cuadrado válido no necesariamente es posible construir un cuadrado (n, k) para cualquier k. Por ejemplo, para el 7-dominó la suma de los lados siempre será menor que 90, y por lo tanto no es posible formar un cuadrado $(7, 90)$.

A continuación se muestra la imagen de un cuadrado formado con un 7-dominó donde la suma de cada lado es 45, es decir, es un cuadrado $(7, 45)$. Notar que los números en las esquinas deben ser considerados dos veces. Por ejemplo, el 3 de la ficha 3–4 es contabilizado tanto en la suma del lado superior como en la suma del lado derecho. Este es uno de los cuadrados $(7, 45)$ posibles. En general, para un n y un k existen varias formas de construir un cuadrado (n, k).



Un cuadrado $(7, 45)$. Notar que cada lado suma 45.

Dado un valor n y un valor k el objetivo de este problema es construir un cuadrado (n, k). Para esto, tendrás que implementar algunas funciones que simplificarán enormemente la resolución del problema.

Subtarea 1 (25 pts)

La primera subtarea consiste en implementar la función `cuadrado` que determina para un valor n si es posible construir un cuadrado válido con un n-dominó. En esta subtarea no importa la suma de los valores de cada lado, sólo se pide determinar si es posible construir un cuadrado válido **usando todas las fichas del n-dominó**.

- `bool cuadrado(int n)`
  - `n`: indica el tamaño del n-dominó.
  - `return`: la función debe retornar `true` si es posible construir un cuadrado válido o `false` en caso contrario.

Restricciones

- Para esta subtarea se probará la función `cuadrado` con varios casos donde $0 < n \leq 1000$ (25pts).

Subtarea 2 (35 pts)

La subtarea 2 consiste en implementar la función `verificar` que dada la especificación de un cuadrado construido con fichas de un n-dominó debe verificar que todos los lados de este sumen lo mismo. La función también debe verificar que todas las fichas que conforman el n-dominó sean usadas, es decir, no hay fichas que falten o que se repitan. En esta subtarea para el n-dominó entregado siempre será posible formar un cuadrado válido, es decir, `cuadrado(n)` retorna `true`.

- `bool verificar(int n, int f, int fichas[])`
  - `n`: indica el tamaño del n-dominó.
  - `f`: indica la cantidad de fichas que conforman el n-dominó.
  - `fichas`: arreglo de enteros de tamaño $2 \times$ `f` donde se especifica el cuadrado que hay que verificar. Más adelante se detalla el formato en que el cuadrado es especificado.
  - `return`: la función debe retornar `true` si todos los lados del cuadrado especificado suman lo mismo y todas las fichas que conforman el n-dominó son utilizadas. En caso contrario debe retornar `false`.

Restricciones

- Para esta subtarea se probará la función `verificar` con varios casos donde $0 < n \leq 1000$ (35pts).

Subtareas 3 y 4 (40 pts)

Para las subtareas 3 y 4, debes implementar la función `construir` que recibe un valor n y un valor k, y construye un cuadrado (n, k) en caso de ser posible. En esta función parar el n-dominó entregado siempre será posible formar un cuadrado válido, es decir, `cuadrado(n)` retorna `true`. No obstante, puede ser el caso que para el valor de k no sea posible formar un cuadrado (n, k). Si es posible formar un cuadrado (n, k) la función debe retornar `true`, en caso contrario debe retornar `false`.

- `bool construir(int n, int k, int f, int fichas[])`
  - `n, k`: valores que indican el cuadrado (n, k) que se quiere construir.
  - `f`: indica la cantidad de fichas que conforman un n-dominó.
  - `fichas`: arreglo de enteros de tamaño $2 \times$ `f` donde debes guardar la especificación del cuadrado (n, k). Más adelante se detalla la forma de especificar el cuadrado. Tu función debe llenar este arreglo con la descripción de algún cuadrado (n, k). Su contenido será revisado sólo en el caso de que la función retorne `true`.
  - `return`: la función debe retornar `true` si es posible formar un cuadrado (n, k) o `false` en caso contrario.

Restricciones

- Para la subtarea 3 se probará la función `verificar` con varios casos donde $n = 7$ (20pts).
- Para la subtarea 4 se probará la función `verificar` con varios casos donde $7 < n \leq 1000$ (20pts).

Formato del arreglo *fichas*

La función `verificar` recibe como parámetro un arreglo de enteros `fichas` donde se especifica un cuadrado construido con las fichas de un n-dominó. De la misma forma `construir` debe llenar el arreglo `fichas` para especificar el cuadrado que construye. A continuación se detalla la forma en que un cuadrado es especificado en el arreglo `fichas`.

Si f es la cantidad de fichas que conforman un n-dominó, el arreglo `fichas` es de tamaño $2 \times f$. Por ejemplo, un cuadrado construido con un 7-dominó debe ser especificado en un arreglo de tamaño 56, pues un 7-dominó está conformado por 28 fichas. En el arreglo una ficha es almacenada en dos posiciones consecutivas. La posición 0 y la 1 corresponden a la primera ficha, la 2 y la 3 a la segunda ficha y así hasta la última ficha que corresponde a las posiciones $2 \times f - 2$ y $2 \times f - 1$ en el arreglo. Las fichas en el arreglo se especifican en sentido horario partiendo desde la esquina superior izquierda. Por ejemplo, el cuadrado (7,45) mostrado en la figura de más arriba puede ser almacenado en un arreglo de la siguiente manera.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
6	2	0	4	5	1	4	6	3	1	1	0	4	5	3	4	0	0	...	1	4	5	5	2	3	3	0	0	6	0	2	6	5	2	2	5	3

Detalles de implementación

Debes enviar exactamente un archivo, llamado `domino.cpp`. Este archivo debe contener todas las funciones descritas. Si solo has implementado una de las funciones las otras deben también estar presente y puedes dejarlas vacías. El archivo `domino.cpp` debe incluir el header `domino.h`.

Grader de Ejemplo

Se provee un *grader* junto con algunos archivos de prueba para que puedas testear tu solución. El grader lee de la entrada estándar en el siguiente formato.

La primera línea contiene un entero T correspondiente a la subtarea que se quiere resolver. El número T puede ser 1, 2, 3 o 4.

- Si T es 1, solo sigue una línea, que contiene un entero n correspondiente al tamaño del n-dominó.
- Si T es 2, siguen dos líneas. La primera contiene dos enteros n y f, correspondientes al tamaño y el número de fichas del n-dominó respectivamente. La segunda línea contiene $2 \times f$ enteros entre 0 y n, los valores del arreglo `fichas` en el formato descrito anteriormente.
- Si T es 3 o 4, sigue una sola línea. Esta línea contiene tres enteros n, k y f, donde n indica el tamaño del n-dominó, k indica cuanto deben sumar los lados del cuadrado que se pide construir, y f es el número de fichas que conforman el n-dominó.