# Programozási feladat: Domino

Írjunk dominó játékost C programozási nyelven! Kezdetben egyetlen dominó található az asztalon (input fájl első sora). Ehhez a kezdeti dominóhoz a fájlból beolvasott dominó-szett (input fájl második-harmadik sora) elemeit próbáljuk meg hozzáépíteni.

## A bemenet formátuma

Az input fájl mindig három sorból áll. Az elsőben a kezdő dominó számpárja, a másodikban a domino szett darabszáma, míg a harmadikban a teljes domino szett számai szerepelnek egy-egy szóközzel elválasztva (az első két szám az első dominót, a harmadik-negyedik szám a következő dominót írja le stb.).

Például a következő input esetén a kezdő dominó bal oldalán 2, jobb oldalán 6 pöttyöt tartalmaz. 4 további dominó van a szettben, melyek közül az elsőnek a bal oldalán 6, jobb oldalán 3 pötty, a másodiknak bal oldalán 3, jobb oldalán 1 pötty van.

```
2 6
4
6 3 3 1 1 2 2 2
```

## Reprezentáció

- A dominókat két számmal reprezentáljuk: ezek megadják a dominó két felén lévő pöttyök számát.
  Elvárás: használj struct típust egy-egy dominó leírására.
- A dominószettet tömbben tároljuk, ahol a tömb elemei lehetnek az egyes dominók, vagy azokra mutató mutatók.
  Elvárás: dinamikusan foglalj pontosan annyi helyet amennyi a beolvasott dominók tárolásához szükséges. Változó hosszúságú tömb (VLA) nem használható.

## A játék menete

- A dominókat addig próbáljuk letenni a meglévő sor (kezdetben egyetlen dominó) jobb vagy bal végére, amíg el nem fogy a szett, vagy el nem akadunk megfelelő dominó hiánya miatt. Ha a sor mindkét végére illeszthető dominó, a jobb oldalra kell illeszteni.
- Minden körben egy dominót próbálunk letenni. Egy dominó akkor letehető, ha valamelyik oldalán az a szám szerepel, ami az asztalon lévő sor jobb vagy bal végén van. A dominók forgathatóak, tehát egy 1|3 dominó 3|1 elhelyezéssel is letehető.
- Ha több letehető dominó van, mindig a szettben legkorábban szereplőt kell választani. Ehhez érdemes a szettet mindig balról-jobbra ellenőrizni, és a legelső letehető dominót választani.
  Tipp: használj dupla ciklust, a külső feleljen egy dominó elhelyezésért, a belső pedig ennek megkeresésért a szettben.
- Ha egy dominót letettünk a sorba, többször már nem használhatjuk fel. Erre ügyelj a megoldásodban.
  Tipp: dominó struct mutatók tömbjeként tárolhatod a dominó szettet, így amikor valamelyik dominót letetted az asztalra, NULL-ra tudod állítani a tömbben a dominó mutatóját.
- A dominókból egyszerű sort építünk, nincs lehetőség kanyarodásra, elágazásra, egymás mellé tételre.
  Tipp: Elég, ha a sor jobb és bal végén aktuálisan szereplő számokat "jegyzed meg", majd frissíted, ha leraktál egy dominót.

## A kimenet

Az összes lerakott dominót ki kell írni (beleértve a kezdeti dominót is). Habár a dominókat elforgatva is letehetjük az asztalra, a képernyőre íráskor mindig az eredeti állásban írjuk ki őket (a könnyebb azonosíthatóság érdekében). A pontos kimeneti formátumot a példa kimenetekből ki tudod olvasni.

## További megkötések

Elvárás: a megoldásod egyetlen forrásfájlba rendezd, de a main()-en kívül legyen legalább még egy függvényed, amit használsz is az eredmény kiszámításához.
Tipp: lehet egy függvényed annak ellenőrzésére, hogy egy konkrét dominó passzol-e egy másikhoz (gyakorlatban az asztalon lévő sor egyik vagy másik végéhez).

## Segítség a fájlból olvasáshoz

Emlékeztetőül, a következőképp lehet fájlból egész számot beolvasni C-ben:

```
int a;
FILE* fp = fopen ("input01.txt", "r");
fscanf(fp, "%d", &a);
```

## Példa bemenetek és kimenetek

Az összes példafájl elérhető a Canvas Fájlok között (**exam0524** mappa).

*Egyszerű példa (input01.txt)*

```
2 6
4
6 3 3 1 1 2 2 2
```

Output:

```
Initial domino: 2|6
Added to the right end: 6|3
Added to the right end: 3|1
Added to the right end: 1|2
Added to the right end: 2|2
Summary: 4 domino were placed.
```

Szépen sorban haladtunk, mindig pont a szettben következő dominót tudtuk lerakni forgatás nélkül a sor jobb végére.

*Komplikáltabb példa (input04.txt)*

```
2 6
4
6 2 3 1 1 2 2 2
```

Output:

```
Initial domino: 2|6
Added to the right end: 6|2
Added to the right end: 1|2
Added to the right end: 3|1
Added to the left end: 2|2
Summary: 4 domino were placed.
```

Ebben a példában már van többféle lerakási eset: forgatás (1|2 -t 2|1-ként rakjuk le), nem tudjuk mindig a szettben következő dominót letenni (6|2 után a következő lehető az 1|2), illetve teszünk a sor bal végére is.

## Értékelés

A C nyelvű megoldásod 30 pontot ér, ha

- hibaüzenet és warning nélkül lefordul (-std=c99 kapcsolóval)
- helyes eredményt produkál (nem csak a kiadott tesztfájlokra)
- a feladat szövegében jelzett implementációs elvárásoknak maradéktalanul megfelel a megoldás (az implementációs tippek csak segítséget adnak, de nem kötelező a betartásuk)

## Extra pontok

Extra pontokat lehet szerezni a C megoldás továbbfejlesztésével, illetve Python megoldás készítésével.

### C továbbfejlesztése

- Egészítsd ki, módosítsd úgy a C implementációt, hogy a teljes felépített dominó sort balról jobbra kiírod a képernyőre a játék végén.
  Tipp: legyen egy struct domino pointereket tartalmazó tömb, melynek maximális elemszáma kétszerese a domino-szett méretének. A közepére tedd be a legelső dominót (pontosabban egy rá mutató pointert), majd a bal-jobb végeket folyamatosan építsd. A felhasznált dominót ne másold/duplikáld, csak a pointerét vedd ki a domino-szettből és rakd be ebbe az új tömbbe a megfelelő indexű pozícióra.

### Python megoldás készítése

- Adj a fent ismertetett domino feladatra egy Python megoldást, az alábbi egyszerűsítéseket figyelembe véve:
  - egy dominót egy tuple-ként reprezentálj
  - a dominó-szett tuple-ök listája legyen
  - nem kell fájlból olvasni, függvényként írd meg a domino feladatot, mely függvénynek két bemenő paramétere van: az első egy tuple, a kezdeti domino, a második pedig egy lista, melyben a domino-szett elemei vannak
  - a függvényt tesztelési célból beégetett globális változókkal hívd meg