Zárthelyi dolgozat

Határidő 2022. nov 4 20:00-ig **Pont** 50 **Beküldés...** egy fájlfeltöltés **Fájltípusok** zip
Elérhető 2022. nov 4, 16:00 - 2022. nov 4, 20:00 körülbelül 4 óra

A zárthelyi alatt egy programozási feladatot kell megoldani, legfeljebb 50 pontért. A megoldásra 180 perc áll rendelkezésre. A megoldáshoz semmilyen segédeszköz nem használható, kivéve a **C referenciát** (https://en.cppreference.com/w/c) . A megoldást **zip** állományként kell feltölteni.

Ne feledd, legalább 10 pontot el kell érni a zh-n a tárgy sikeres teljesítéséhez. A nem forduló megoldásra automatikusan 0 pont jár.

Táblás amőba

tictactoe.png

A feladat keretében egy kétdimenziós, táblás amőba játékot kell implementálni. A játékban két játékos vehet részt, akik felváltva dobnak egy-egy különböző színű korongot a táblába. A korongok leesnek a tábla adott oszlopában lévő legmélyebb szabad helyére. A játékosok közül az nyer, akinek legalább három korongja vízszintesen vagy függőlegesen egymás mellé vagy alá kerül.

Az oszlopokat nagybetűkkel azonosítjuk: A, B, C, D, E, F, G. Végig ugyanarról az oldalról nézzük a táblát. A korongok száma nincs limitálva, de 6 * 7-nél többet nem lehet legálisan elhelyezni. A játékmenetet egy C-string írja le, pl. *“ABDCAGEEE”*, amit azt jelenti, hogy a kezdőjátékos először a baloldali oszlopba dobott, erre válaszként a második játékos eggyel mellé. Ezután megint a kezdőjátékos jön, aki a D oszlophoz dobja a korongot. Tehát a stringben két karakterenként írhatunk le egy kört.

A programot teszteljük több különböző játékmenettel, pl.

- “ABDCAGEEE” (döntetlen játék)
- “ABDCAEEEEEEFFFAC” (az 1. játékos győz)
- “ABDCAEEEEEEFFG” (a 2. játékos győz)
- “AAAAAAABBBBBBBCCCCCCCDDDDDDDEEEEEEEFFFFFFFGGGGGG” (döntetlen játék 6 hibás lépéssel)

A programot egyetlen *c* kiterjesztésű fájlban kell megírni, és *zip* állományként, tömörítve feltölteni a gyakorlati csoport Canvas felületére.

10 pont

A programban a táblát reprezentálja egy kétdimenziós, egészeket tároló tömb, ahol a tábla magassága (tehát a sorok száma) 6, szélessége (az oszlopok száma) 7. Ezeket a paramétereket vegyük fel globális változóként! A táblán és a méretein kívül más globális változó ne szerepeljen a programban.

Írjunk egy `init` függvényt, amelyet a játék kezdetekor hívunk meg. Ez a függvény üresre állítja a tábla minden “mezőjét”. Egy mező üres, ha az értéke 0.

Legyen a programban egy `printTable` nevű függvény, amely a tábla aktuális állapotát jeleníti meg a képernyőn.

15 pont

Írjuk meg a `submit` függvényt, amely paraméterként megkapja, hogy melyik játékos melyik oszlopba próbál dobni. A játékosokat azonosítsuk különböző egész számokkal, pl. 1 és 2. A gravitáció miatt a lehető legmélyebbre esik a korong a paraméterként kapott oszlopban. Ha nincs szabad hely az oszlopban, vagy nem létező oszlopba próbálunk dobni, akkor jelezzük, hogy a lépés nem legális, és folytassuk a játékot a következő lépéssel. A függvény adja vissza, hogy legális lépés történt-e, és írjon ki hibaüzenetet, ha nem volt legális a lépés.

15 pont

Írjuk meg az `evaluate` függvényt, amely minden lépés után megvizsgálja, hogy nyert-e valamelyik játékos. A nyeréshez függőlegesen vagy vízszintesen 3 azonos színű korong szükséges.

Vegyük fel a `game` függvényt, amely paraméterként megkapja a játék során végrehajtandó dobási sorrendet, és meghatározza, mi a játék kimenete az eddig megírt függvények segítségével.

10 pont

A `main` függvény inicializáltatja a táblát, lejátssza a játékot a `game` segítségével, majd a standard outputra írja a játék eredményét:

- First player wins. (Ha a kezdő játékos nyert.)
- Second player wins. (Ha a másodikként dobó játékos nyert.)
- Draw. (Döntetlen/ Nincs nyertes)

Elvárások a programmal szemben

- A nem forduló kód automatikusan 0 pontot ér. (Természetesen ez csak a legutoljára feltöltött megoldásra vonatkozik.)
- Ne használj globális változókat a tábla adatain kívül!
- Logikusan tagold a megoldást. A megoldás részeit külön függvényekben valósítsd meg.
- Kerüld a nem definiált viselkedést okozó utasításokat!