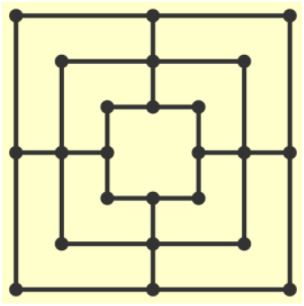
Féléves Feladat

A feladat megoldásaként a teljes solution mappát betömörítve a Moodle rendszerben kell leadni. A beadandó solution elnevezése a féléves feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD.zip**.

A feladat akkor kerül elfogadásra, ha a hallgató azt személyesen megvédte.

A feladattal kapcsolatos további információk az utolsó oldalon találhatók (ezen ismeretek hiányából adódó reklamációt nem fogadunk el!).

Készítsen malom játékot, ahol a felhasználó a gép ellen játszhat.



A malom játék táblája

A szabályok a következők:

- A játékot a képen látható felépítésű táblán játsszák, 9-9 db bábuval. A felhasználó játsszon a kék, a gép pedig a piros színű bábukkal. *(A bábúk jelölése tetszőleges.)*
- A játék első szakaszában a játékosok felváltva, egyesével felrakják a bábuikat a táblára. Ezek után a bábuk mozgatásával igyekeznek "malmot" alkotni (egy sorban vagy oszlopban három azonos színű bábu). Amelyik játékosnak ez sikerül, az leveheti az ellenfél egyik bábuját, de csak olyat, amelyik nem alkot malmot. (Ha már nincs ilyen, akkor bármelyiket le lehet venni.)
- Lépni csak vízszintesen és függőlegesen lehet a vonalak mentén (tehát átlósan nem).
- Az a játékos veszít, aki hét bábuját elvesztette (ilyenkor már nem tud malmot kirakni).
- Ha a gépnek van lehetősége malmot kirakni, akkor megteszi. Ha nincs akkor egy véletlenszerűen kiválasztott bábuval tesz egy szabályos lépést.

Megkötések

- A lépés megadása a bábu helyének és a célmezőnek a beírásával történjen, a mezők jelölése tetszőleges.
- Ha egy megadott lépés nem szabályos, akkor erre figyelmeztessen a program, és kérjen be egy új "lépést".
- A játékban legyen lehetőség a folyamatban lévő játékot menteni a felhasználó által megadott néven, és korábban mentett játékot betölteni és folytatni.

A megoldás során tartsa be a tanult OOP alapelveket (egységbezárás, láthatóság, öröklés), törekedjen saját osztályok létrehozására. Ahol lehetséges alkalmazza a megtanult programozási tételeket, illetve használja a tanult technikákat.

Tájékoztató

A feladattal kapcsolatosan általános szabályok:

- A feladat megoldását egy Console Application részeként kell elkészíteni.
- A feladat megoldásaként beadni vagy a betömörített solution mappa egészét vagy a `Program.cs` forrásfájlt kell (hogy pontosan melyiket, azt minden feladat külön definiálja), melynek elnevezése a feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD**[.zip|.cs]
- A megvalósítás során lehetőség szerint alkalmazza az előadáson és a laboron ismertetett programozási tételeket és egyéb algoritmusokat figyelembe véve a *Megkötések* pontban definiáltakat, ezeket leszámítva viszont legyen kreatív a feladat megoldásával kapcsolatban.
- Az alkalmazás elkészítése során minden esetben törekedjen a megfelelő típusok használatára, illetve az igényes (*formázott, felesleges változóktól, utasításoktól mentes*) kód kialakítására, mely magába foglalja az elnevezésekkel kapcsolatos ajánlások betartását is (bővebben).
- **Ne másoljon vagy adja be más megoldását!** Minden ilyen esetben az összes (felépítésben) azonos megoldás duplikátumként lesz megjelölve és a megoldás el lesz utasítva.
- **Idő után leadott vagy helytelen elnevezésű megoldás vagy a kiírásnak nem megfelelő megoldás vagy fordítási hibát tartalmazó vagy (helyes bemenetet megadva) futásidejű hibával leálló kód nem értékelhető!**
- A feladat leírása az alábbiak szerint épül fel (* - opcionális):
  - *Feladat leírása* - a feladat megfogalmazása
  - *Bemenet* - a bemenettel kapcsolatos információk
  - *Kimenet* - az elvárt kimenettel kapcsolatos információk
  - *Megkötések* - a bemenettel, a kimenettel és az algoritmussal kapcsolatos megkötések, melyek figyelembevétele és betartása kötelező
  - **Megjegyzések* - további, a feladattal, vagy a megvalósítással kapcsolatos megjegyzések
- A leadott megoldással kapcsolatos minimális elvárás:
  - Nem tartalmazhat fordítás idejű figyelmeztetést.
  - Nem tartalmazhat fordítási hibát.
- A feladat megoldásához minden esetben elegendő a ***.NET Framework 4.7.2***, illetve a ***C# 7.3***, azonban megoldását elkészítheti ***.NET 5***-öt, illetve a ***C# 9***-et használva is, viszont a nyelv újjításait nem használhatja. További általános, nyelvi elemekkel való megkötés, melyet a házi feladatok során nem használhat a megoldásában:
  - Methods: `Array.Sort`, `Array.Reverse`, `Console.ReadKey`, `Environment.Exit`
  - LINQ: `System.Linq`
  - Attributes
  - Collections: `ArrayList`, `BitArray`, `DictionaryEntry`, `Hashtable`, `Queue`, `SortedList`, `Stack`
  - Generic collections: `Dictionary<K,V>`, `HashSet<T>`, `List<T>`, `SortedList<T>`, `Stack<T>`, `Queue<T>`
  - Keywords:
    - Modifiers: `protected`, `internal`, `abstract`, `async`, `event`, `external`, `in`, `out`, `sealed`, `unsafe`, `virtual`, `volatile`
    - Method parameters: `params`, `in`, `out`
    - Generic type constraint: `where`
    - Access: `base`
    - Contextual: `partial`, `when`, `add`, `remove`, `init`
    - Statement: `checked`, `unchecked`, `try-catch-finally`, `throw`, `fixed`, `foreach`, `continue`, `goto`, `yield`, `lock`, `break` - *in loop*
    - Operator and Expression:
      - Member access: `^` - *index from end*, `..` - *range*
      - Type-testing: `is`, `as`, `typeof`
      - Conversion: `implicit`, `explicit`
      - Pointer: `*` - *pointer*, `&` - *address-of*, `*` - *pointer indirection*, `->` - *member access*
      - Lambda: `=>` - *expression, statement*

- Others: `?:` - *tenary*, `!` - *null forgiving*, `?.` - *null conditional member access*, `?[]` - *null conditional element access*, `??` - *null coalescing*, `??=` - *null coalescing assignment*, `::` - *namespace alias qualifier*, `await`, `default` - *operator, literal*, `delegate`, `is` - *pattern matching*, `nameof`, `sizeof`, `stackalloc`, `switch`, `with` - *expressiong*, `operator`
- Types: `dynamic`, `interface`, `object`, `Object`, `var`, `struct`, `nullable`, `pointer`, `record`, `Tuple`, `Func<T>`, `Action<T>`,