A	**Név:**	**Neptun kód:**
Gép sorszám:	**Értékelő oktató:**	**Jegy:**

Feladat: Konzultáció

Készítsünk egy *WPF* grafikus felületű, *MVVM* architektúrájú alkalmazást a következő egy személyes játékra. A játékos célja, hogy az általa irányított hallgatóval mozogva minél több egyetemi oktatójával tudjon konzultálni még a vizsgaidőszak előtt, eredményesebb felkészülést biztosítva a vizsgákra. Az oktatók azonban csak ritkán – és véletlenszerűen – találhatóak meg a szobáikban, így a hallgató feladata nem egyszerű.

A játékot egy egyetemi folyosót reprezentáló $5 \times n$-es táblán játszák, n értéke páratlan. Kezdetben a hallgató (H) a folyosó, azaz a tábla közepén helyezkedik el, az oktatói szobák ajtajai (A) pedig a folyosó két szélén (a tábla alján és tetején) minden második mezőt foglalnak el, az ábrán látható módon. Az oktatói szobaajtók lehetnek nyitott vagy zárt állapotban, amely állapotukat folyamatosan változtatják, ugyanis az oktatók rendszeresen ki-be járkálnak a szobáikból. A hallgató célja, hogy adott idő alatt minél több nyitott szobaajtóra rálépjen.

A		A		A		A
			H			
A		A		A		A

Részfeladatok:

1. **(2 pont)** A program jelenítse meg a játéktáblát, amelynek méretét a felhasználó adja meg három lehetőségből választva (kicsi: 5×7, közepes: 5×9, vagy nagy: 5×11). A program 2 másodpercenként változtassa véletlenszerűen valahány, de legfeljebb $(n-1)/2$ zárt ajtó állapotát nyitottra. Az ajtók aktuális állapota vizuálisan is kerüljön megjelenítésre (pl. eltérően színezve). A kinyitott ajtók 3 másodperc múlva automatikusan záródjanak be. A játékosnak legyen lehetősége a hallgatóval a táblán lépegetni balra, jobbra, lefele és felfele. A hallgató bármely mezőre ráléphet, amennyiben egy nyitott ajtóra lép, akkor az azonnal bezárul (sikeres konzultáció).

2. **(1 pont)** Egy játék a programmal tartson 30 másodpercig, azonban minden sikeres konzultációval (amennyiben a hallgató nyitott ajtóra lépett) nyerjen plusz 5 másodperc játékidőt a játékos. Az alkalmazás jelenítse meg hány másodperc játékidő van még hátra és ismerje fel, ha vége van a játéknak (letelt az idő). Ekkor jelenítse meg, hogy összesen hány konzultációt tudott a hallgató végrehajtani, majd automatikusan kezdődjön új játék. Lehessen továbbá bármikor új játékot kezdeni, akár a táblaméret megváltoztatásával.

3. **(1 pont)** Az alkalmazásban legyen lehetőség a 10 legtöbb pontot elért játék toplistájának megtekintésére (játék időpontja, konzultációk száma). Ehhez minden játék végén perzisztens módon relációs adatbázisban kerüljön tárolásra az adott játék befejezési időpontja és az elért pontszám (konzultációk száma). Az adatbáziskezelést objektum-relációs leképezéssel, *Entity Framework* használatával végezd, a mentésért és betöltésért külön *perzisztencia* réteg feleljen.

4. **(1 pont)** Az alkalmazás minden játék végén kérje be a felhasználó nevét és ezt is tárolja el a szerzett pontszámhoz. A toplistában jelenjenek meg a nevek is.

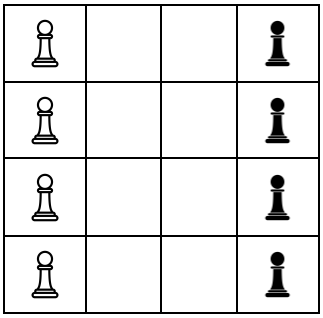
A megoldást az elfogadást követően **`ZIP`** formátumban fel kell tölteni a **`https://assignment.elte.hu/`** beadandókezelő rendszerbe.

Jó munkát!

B	**Név:**	**Neptun kód:**
Gép sorszám:	**Értékelő oktató:**	**Jegy:**

Feladat: Áttörés

Készítsünk *WPF* grafikus felületű, *MVVM* architektúrájú alkalmazást a következő két személyes játékra. A játékot egy $n \times n$-es táblán játsszák, kezdetben *n* - *n* gyaloggal, amelyek egymással szemben helyezkednek el (a tábla bal, illetve jobb oldali oszlopaiban). A gyaloggal a sakkban ismert módon lehet lépni, azaz csak előre léphet, és átlósan üthet. A játékosok egyenként lépegetnek, illetve ütnek. A játék célja, hogy egy gyaloggal átérjük a túloldalra.



Részfeladatok:

1. **(2 pont)** A program jelenítse meg a játéktáblát, amelynek méretét a felhasználó adja meg három lehetőségből választva (4×4 , 6×6 , vagy 8×8). A játéktáblán jelenjenek meg a figurák is a tábla bal, illetve jobb szélső oszlopában. A játékosoknak legyen lehetősége egymást váltva lépegetni, illetve ütni egy bábuval, amit először ki kell választania. A program csak szabályos lépéseket tegyen lehetővé, folyamatosan jelezze a képernyőn, melyik játékos következik, és hogy kijelölést, vagy lépést kell-e végeznie. Minta figurák: **`http://mcserep.web.elte.hu/pawn.zip`**

2. **(1 pont)** A program ismerje fel, ha vége a játéknak, ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdődjön új játék. Lehessen továbbá bármikor új játékot kezdeni, akár a táblaméret megváltoztatásával.

3. **(1 pont)** A program kövesse a játékidőt, és egy lépéshez legfeljebb 10 másodpercet adjon. Amennyiben a játékos ennyi idő alatt nem lép, akkor automatikusan a másik játékos következik. A program jelenítse meg folyamatosan a fennmaradó játékidőt is.

4. **(1 pont)** Legyen lehetőség a játékállás elmentésére, valamint betöltésére. Játék betöltéskor a soron következő játékos helyesen kerüljön visszaállításra és a lépéséből hátralévő játékidő is onnan folytatódik, ahol elmentettük. A mentés relációs adatbázisba történjen perzisztens módon, az adatbáziskezelést objektum-relációs leképezéssel, *Entity Framework* használatával végezd, a mentésért és betöltésért külön *perzisztencia* réteg feleljen.

A megoldást az elfogadást követően **`ZIP`** formátumban fel kell tölteni a **`https://assignment.elte.hu/`** beadandókezelő rendszerbe.

Jó munkát!