1. beadandó feladat: egyszerű grafikus felületű alkalmazás

Közös követelmények:

- A megvalósításnak felhasználóbarátnak, és könnyen kezelhetőnek kell lennie. A szerkezetében törekedni kell az objektumorientált szemlélet megtartására (a főprogram kivételével a szerkezetnek teljesen objektumorientáltnak kell lennie, de nem kötelező a nézet és a modell szétválasztása).
- A megjelenítéshez lehet vezérlőket használni, vagy elemi grafikát. Egyes feladatoknál különböző méretű játéktábla létrehozását kell megvalósítani, ekkor ügyelni kell arra, hogy az ablakméret mindig alkalmazkodjon a játéktábla méretéhez.
- A dokumentációnak tartalmaznia kell a feladat elemzését, felhasználói eseteinek (WHEN-GIVEN-THEN szerkezetű) leírását (UML felhasználói esetek diagrammal kiegészítve), a program statikus szerkezetének leírását (UML osztálydiagrammal), valamint az esemény-eseménykezelő párosításokat és az eseménykezelő tevékenység rövid leírását.

Feladatok:

1. **Hanoi tornyai**

   Készítsünk programot, amellyel a közismert Hanoi tornyai játékot lehet játszani. Adott három rúd, és az első rúdon n korong, amelyek alulról felfelé egyre kisebb méretűek.
   A játék célja, hogy az összes korongot helyezzük át az első rúdról a másodikra úgy, hogy minden lépésben csak egy korongot mozgathatunk, és egy korongot mindig csak egy nála nagyobb korongra vagy üres rúdra helyezhetünk. A programban a korongok áthelyezése történjen úgy, hogy először kijelöljük azt a rudat, amelyikről a legfelső korongot mozgatni akarjuk, aztán pedig azt a rudat, amelyikre át akarjuk tenni a korongot. A program csak a szabályos áthelyezéseket engedélyezze.
   A program biztosítson lehetőséget új játék kezdésére a korongok számának megadásával (3, 5, 8), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel (áthelyezéssel) győzött a játékos, majd kezdjen automatikusan új játékot.

2. **Misszionárius-kannibál**

   Készítsünk programot, amely bemutatja a misszionárius-kannibál problémát.
   Adott egy folyó, amelynek az egyik partján n darab kannibál és n darab misszionárius várakozik, hogy átkeljenek. Átkelésükhöz adott továbbá egy csónak, amely maximum k személyt tud egyszerre szállítani. Az átkelések

alkalmával nem szabad, hogy egy időben akár a csónakban, akár valamelyik parton több kannibál legyen, mint misszionárius, mivel akkor megeszik a misszionáriusokat (kivéve, ha ott egyetlen misszionárius sem tartózkodik).
Jelenítsük meg a két parton és a csónakban lévő misszionáriusokat és kannibálokat. Lehessen a két partról a csónakba, valamint visszahelyezni bárkit, illetve kezdeményezni átkelést, amely csak akkor történik meg, ha a játékállás a feltételeket az átkelés után is kielégíti.
A program biztosítson lehetőséget új játék kezdésére n és k értékének megadásával (2-től 5-ig), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel (átkeléssel) győzött a játékos, majd kezdjen automatikusan új játékot.

3. Átkelés a hídon

Készítsünk programot, amellyel az utazó kereskedők problémáját mutatjuk be.
Ebben néhány kereskedő éjszaka ér egy mély szakadék felett átívelő rozoga hídhoz. Ezen a hídon a sötétben csak lámpával lehet biztonságosan átkelni, de az utazóknak sajnos mindössze egyetlen lámpájuk van, továbbá egyszerre legfeljebb hárman juthatnak át, és valakinek vissza kell menni a lámpával a többiekért. A kereskedők különböző korúak (fiatal, középkorú, idős), ezért eltérő idő kell nekik a hídon való átkeléshez. Természetesen, amikor többen mennek át egyszerre, akkor a lassúbbhoz kell igazítani a lépést. Jelenítsük meg a szakadék két partján lévő kereskedőket, biztosítsuk azt, hogy mindig felváltva tudjunk egyik vagy másik partról 1-3 személyt kiválasztani, amely átkerül majd a másik oldalra. A program folyamatosan számolja az átkelés idejét (természetesen nem valós időben, hanem az előre megadott átkelési idők segítségével).
A program biztosítson lehetőséget új játék kezdésére a fiatal, középkorú és idős kereskedők számának megadásával (0-tól 5-ig, minimum 3 különböző összeállításból választva), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, milyen idővel győzött a játékos, majd kezdjen automatikusan új játékot.

4. Tili-toli

Készítsünk programot, amellyel a következő játékot lehet játszani.
Adott egy $n \times n$ mezőből tábla, amelyen véletlenszerűen elhelyezünk $n^2 - 1$ számozott bábut (1, 2, ..., $n^2 - 1$), egy mezőt pedig üresen hagyunk. A játékos feladata az, hogy a bábuk tologatásával kirakjuk a "sorfolytonos" sorrendet, vagyis a számok sorban következzenek az első sorban balról jobbra, majd a második sorban $(n + 1)$-től indulva balról jobbra, és így tovább. A tologatások során egy bábút áthelyezhetünk az egyetlen üres mezőre, ha azzal szomszédos mezőn áll (csak vízszintesen és függőlegesen lehet mozogni, átlósan nem).

A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (3×3, 4×4, 6×6), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel győzött a játékos, majd kezdjen automatikusan új játékot.

5. Rubik óra

Készítsünk programot, amellyel egy Rubik órát lehet kirakni.
A játéktáblán helyezzünk el 9 darab órát 3 sorba és 3 oszlopba rendezve, mindegyik 1-12 közötti értéket mutasson (kezdetben véletlenszerű beállítással). Azokon a helyeken, ahol 4 óra sarka is összeér (összesen 4 ilyen pozíció van) legyen egy-egy nyomógomb is a táblán úgy, hogy ezek a szomszédos 4 óra állását tudják eggyel növelni. (Tehát 4 óra van, amit csak egy gomb növel, 4, amit kettő, és 1, amit mind a négy gomb növel). A játék célja az, hogy a gombokkal történő állítással mind a 9 óra 12-t mutasson.
A program biztosítson lehetőséget új játék kezdésére, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel (állítással) győzött a játékos, majd kezdjen automatikusan új játékot.

6. Rubik tábla

Készítsünk programot, amellyel egy Rubik táblát lehet kirakni.
A Rubik tábla lényegében a Rubik-kocka két dimenziós változata. A játékban egy $n \times n$ mezőből álló táblán n különböző színű mező lehet, mindegyik színből pontosan n darab, kezdetben véletlenszerűen elhelyezve. A játék célja az egyes sorok, illetve oszlopok mozgatásával (ciklikus tologatásával, azaz ami a tábla egyik végén lecsúszik, az ellentétes végén megjelenik) egyszínűvé alakítani vagy a sorokat, vagy az oszlopokat (azaz vízszintesen, vagy függőlegesen csíkokat kialakítani).
A program biztosítson lehetőséget új játék kezdésére a táblaméret (és így a színek számának) megadásával (2×2, 4×4, 6×6), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel győzött a játékos, majd kezdjen automatikusan új játékot.

7. Királynők

Készítsünk programot, amellyel a következő játékot lehet játszani.
Adott egy $n \times n$-es tábla, melyen királynőket helyezhetünk el sorban egymás után. A tábla kezdetben üres, és a játék célja, hogy elhelyezzünk n királynőt úgy, hogy azok közül semelyik kettő ne üsse egymást (vízszintesen, függőlegesen, vagy átlósan). Minden elhelyezés után jelöljük meg a táblán azokat a mezőket, ahova már nem rakhatunk újabb királynőt (amelyeket az eddig elhelyezett bábúk ütnek), és természetesen ne is engedjük ezeket a mezőket használni. A

lehelyezett királynőt lehessen visszavenni, ekkor a program szabadítsa fel a megfelelő mezőket.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (4×4, 6×6, 8×8), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel győzött a játékos (a levételek is lépésnek számítanak), majd kezdjen automatikusan új játékot.

8. Lovagi torna

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelynek a négy sarkába 2-2 fehér, illetve fekete ló figurát helyezünk el (az azonos színűek ellentétes sarokban kezdenek).
A játékosok felváltva lépnek, a figurák L alakban tudnak mozogni a játéktáblán. Kezdetben a teljes játéktábla szürke színű, de minden egyes lépés után az adott mező felveszi a rá lépő figura színét (bármilyen színű volt előtte). A játék célja, hogy valamely játékosnak függőlegesen, vízszintesen, vagy átlósan egymás mellett 4 ugyanolyan színű mezője legyen. Ha ezt valamelyik játékos elérte, vége a játéknak.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (4×4, 6×6, 8×8), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

9. Áttörés

Készítsünk programot, amellyel a következő kétszemélyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, ahol a két játékos bábúi egymással szemben helyezkednek el, két sorban (pont, mint egy sakktáblán, így mindkét játékos $2n$ bábuval rendelkezik, ám mindegyik bábu ugyanolyan típusú). A játékos bábúival csak előre léphet egyenesen, vagy átlósan egy mezőt (azaz oldalra, és hátra felé nem léphet), és hasonlóan ütheti a másik játékos bábúját előre átlósan (egyenesen nem támadhat). Az a játékos győz, aki először átér a játéktábla másik végére egy bábuval.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (6×6, 8×8, 10×10), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

10. 4-es játék

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.
Adott egy $n \times n$ mezőből álló tábla, amelynek mezői 0 és 4 közötti értékeket tartalmaznak. Kezdetben minden mezőn a 0 érték van. Ha a soron következő játékos a tábla egy tetszőleges mezőjét kiválasztja, akkor az adott mezőn és a szomszédos négy mezőn az aktuális érték eggyel nő felfelé, ha az még kisebb,

mint 4. Aki a lépésével egy, vagy több mező értékét 4-re állítja, annyi pontot kap, ahány mezővel ezt megtette. A játékosok pontjait folyamatosan számoljuk, és a játékmezőn eltérő színnel jelezzük, hogy azt melyik játékos billentette 4-esre. A játék akkor ér véget, amikor minden mező értéke 4-et mutat. Az győz, akinek ekkor több pontja van.

A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (3×3, 5×5, 7×7), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.