3. beadandó feladat: grafikus felületű játékprogram

Közös követelmények:

- A megvalósításnak működő programot kell biztosítania. A program elindítását követően egyértelműnek kell lennie, melyik feladatot oldja meg (szöveges információk figyelembe vétele nélkül). A program működése során csak a grafikus képernyőt használhatja (konzolt nem), és billentyűzettel, illetve egérrel lehet vezérelni. A programból bármikor ki lehessen lépni az ESC billentyű segítségével.

- A megvalósításban törekedni kell az objektumorientált szemléletmód követésére, a programnak teljesen objektumorientáltnak kell lennie. Tartalmaznia kell a főprogramot (amely 3 soros lehet), az alkalmazás és a vezérlők osztályait, valamint a játék működéséhez szükséges további osztályokat.

- A grafikus felületet vezérlőkből építsük fel, amelyek egy közös ős osztályból származnak. A megvalósításban használjuk fel a korábban elkészített (kiválasztó, számbeállító) vezérlőket, pl. a lépésszám, játékméret kiválasztásánál. Felhasználható továbbá bármilyen, az előadás/gyakorlat során publikussá tett vezérlő (pl. címke, gomb).

- A kód legyen megfelelően kommentezett. A főprogram kódfájlja a hallgatói adataival, illetve a program kezelésével kapcsolatos leírással kezdődjön.

- A feltűntetett pontszámok az alapjáték megvalósítására adható pontszámok. További funkciók, kiegészítések, illetve minőségi megoldások növelhetik a maximális pontszámot.

Feladatok:

1. Hanoi tornyai (20 pont)

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.

2. Tili-toli (20 pont)

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.

3. Misszionárius-kannibál (20 pont)

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.

4. Átkelés a hídon (20 pont)

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.

5. Királynők (20 pont)

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).

6. Áttörés (30 pont)

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.

7. 4-es játék (30 pont)

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.

8. Lovagi torna (30 pont)

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. A játéknak akkor van vége, ha minden mező kapott valamilyen színt.
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.

9. Kitolás (30 pont)

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelyen kezdetben a játékosoknak n fehér, illetve n fekete kavics áll rendelkezésre, amelyek elhelyezkedése véletlenszerű.
A játékosok kiválaszthat egy saját kavicsot, amelyet függőlegesen, vagy vízszintesen eltolhat. Eltoláskor azonban nem csak az adott kavics, hanem a vele az eltolás irányában szomszédos kavicsok is eltolódnak, a szélső mezőn lévők pedig lekerülnek a játéktábláról. A játék célja, hogy adott körszámon belül ($5n$) az ellenfél minél több kavicsát letoljuk a pályáról (azaz nekünk maradjon több kavicsunk a végére). Ha mindkét játékosnak ugyanannyi marad, akkor a játék döntetlen.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 4×4, 6×6) és így a lépésszám (15, 20, 30) megadásával, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött (ha nem döntetlen).

10. Rubik tábla (30 pont)

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.

11. Négyzetek (30 pont)

Készítsünk programot, amellyel az alábbi két személyes játékot játszhatjuk.
Adott egy $n \times n$ pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre akár két négyzet is elkészülhet. A játék addig tart, amíg lehet húzni vonalat a táblán.
A játékosok felváltva húzhatnak egy-egy vonalat, de ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 9×9). Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen). Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

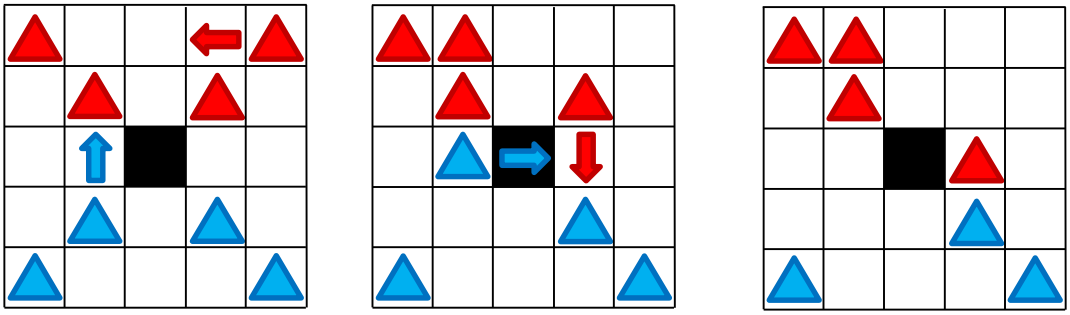
12. Vadászat (30 pont)

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.
Adott egy $n \times n$ mezőből álló tábla, ahol egy menekülő és egy támadó játékos helyezkedik el.
Kezdetben a menekülő játékos figurája középen van, míg a támadó figurái a négy sarokban helyezkednek el. A játékosok felváltva lépnek. A figurák vízszintesen, illetve függőlegesen mozoghatnak 1-1 mezőt, de egymásra nem léphetnek. A támadó játékos célja, hogy adott lépésszámon ($4n$) belül bekerítse a menekülő figurát, azaz a menekülő ne tudjon lépni.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 5×5, 7×7) és így a lépésszám (12, 20, 28) megadásával, folyamatosan jelenítse meg a lépések számát, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött.

13. Fekete lyuk (30 pont)

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelyen két játékos űrhajói helyezkednek el, középen pedig egy fekete lyuk. A játékos $n - 1$ űrhajóval rendelkezik, amelyek átlóban helyezkednek el a táblán (az azonos színűek egymás mellett, ugyanazon az oldalon).

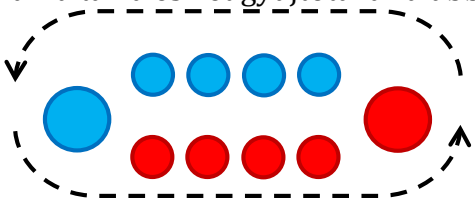
A játékosok felváltva léphetnek. Az űrhajók vízszintesen, illetve függőlegesen mozoghatnak a táblán, de a fekete lyuk megzavarja a navigációjukat, így nem egy mezőt lépnek, hanem egészen addig haladnak a megadott irányba, amíg a tábla széle, a fekete lyuk, vagy egy másik, előtte lévő űrhajó meg nem állítja őket (tehát másik űrhajót átlépni nem lehet). Az a játékos győz, akinek sikerül űrhajóinak felét eljuttatnia a fekete lyukba.



A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (5×5, 7×7, 9×9), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött.

14. Awari (30 pont)

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Két játékos egymással szemben helyezkedik el, közöttük pedig (paraméterként megadható) páros számú tálka és két gyűjtőtál az alábbi lerendezésben.



(8 tálkával)

Mindkét játékos a hozzá közelebbi tálkákat és a tőle jobb kézre eső gyűjtőtálat mondhatja sajátjának. (Így az ellenfél gyűjtőtálja baloldalra esik.) Kezdetben mindegyik tálkában 6-6 kavics van, a gyűjtőtálak pedig üresek.

A játékban a soron következő játékos kiválasztja egyik saját tálkáját (ez nem lehet a gyűjtőtál), hogy azt kiürítse úgy, hogy tartalmát az óramutató járásával ellentétes irányban egyesével beledobálja, majd ismét a saját tálkáiba, de azt a tálkát kihagyva, amelyiknek a kiürítését végezzük, amíg el nem fogynak a

kavicsok. Ha az egyik játékos rákattint valamelyik tálkájára, akkor a tálkában lévő kavicsok áthelyezése automatikusan történjen meg. Ha az utolsó kavics a játékos saját üres tálkáinak egyikébe kerül, akkor ezt a kavicsot, valamint a szemközti tálka tartalmát a saját gyűjtőládába teszi. Viszont, ha az utolsó kavics a játékos saját gyűjtőtálkájába esik, akkor újra ő következik, de ezt csak egyszer teheti meg, hogy ellenfele is szóhoz juthasson.
A játéknak akkor van vége, ha az egyik térfél kiürült, azaz az egyik játékos tálkái mind kiürülnek. Ekkor az a játékos nyeri a játékot, akinek a gyűjtőtáljában több kavics van.
A program biztosítson lehetőséget új játék kezdésére a tálkák számának megadásával (4, 8, 12), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött.

15. Labirintus (30 pont)

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amely labirintusként épül fel, azaz fal, illetve padló mezők találhatóak benne, illetve egy kijárat a jobb felső sarokban. A játékos célja, hogy a bal alsó sarokból indulva minél előbb kijusson a labirintusból.
A labirintusban nincs világítás, csak egy fáklyát visz a játékos, amely a 2 szomszédos mezőt világítja meg (azaz egy 5×5-ös négyzetet), de a falakon nem tud átvilágítani.
A játékos figurája kezdetben a bal alsó sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán.
A pályák méretét, illetve felépítését (falak, padlók) tároljuk fájlban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos), továbbá ismerje fel, ha vége a játéknak. A program játék közben folyamatosan jelezze ki a játékidőt.

16. Maci Laci (30 pont)

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amelyben Maci Lacival kell piknikkosarakra vadásznunk. A játékpályán az egyszerű mezők mellett elhelyezkednek akadályok (pl. fa), valamint piknikkosarak. A játék célja, hogy a piknikkosarakat minél gyorsabban begyűjtsük.
Az erdőben vadőrök is járőröznek, akik adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen). A járőrözés során egy megadott irányba haladnak egészen addig, amíg akadályba (vagy az erdő szélébe) nem ütköznek, ekkor megfordulnak, és visszafelé haladnak (tehát folyamatosan egy vonalban

járőröznek). A vadőr járőrözés közben a vele szomszédos mezőket látja (átlósan is, azaz egy 3 × 3-as négyzetet).
A játékos kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, a piknikkosárra való rálépéssel pedig felveheti azt. Ha Maci Lacit meglátja valamelyik vadőr, akkor a játékos veszít.
A pályák méretét, illetve felépítését (piknikkosarak, akadályok, vadőrök kezdőpozíciója) tároljuk fájlban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Ismerje fel, ha vége a játéknak, és jelezze, győzött, vagy veszített a játékos. A program játék közben folyamatosan jelezze ki a játékidőt, valamint a megszerzett piknikkosarak számát.

17. Menekülj (30 pont)

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, ahol a játékos két üldöző elől próbál menekülni, illetve próbálja őket aknára csalni.
Kezdetben a játékos játékpálya felső sorának közepén helyezkedik el, a két üldöző pedig az alsó két sarokban. Az ellenfelek adott időközönként lépnek egy mezőt a játékos felé haladva úgy, hogy ha a függőleges távolság a nagyobb, akkor függőlegesen, ellenkező esetben vízszintesen mozognak a játékos felé.
A pályán véletlenszerű pozíciókban aknák is elhelyezkednek, amelyekbe az ellenfelek könnyen beleléphetnek, ekkor eltűnnek (az akna megmarad).
A játékos vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, és célja, hogy az ellenfeleket aknára csalja, miközben ő nem lép aknára. Ha sikerül minden üldözőt aknára csalnia, akkor győzött, ha valamely ellenfél elkapja (egy pozíciót foglal el vele), vagy aknára lép, akkor veszített.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (11 × 11, 15 × 15, 21 × 21), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet senki). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy győzött, vagy veszített-e a játékos. A program játék közben folyamatosan jelezze ki a játékidőt.