Programozási technológia I. – 2. beadandó feladatsor

Közös követelmények:

- A megvalósításnak felhasználóbarátnak és könnyen kezelhetőnek kell lennie. Törekedni kell az objektumorientált megoldásra, de nem kötelező a többrétegű architektúra alkalmazása
- 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 eseteit (UML felhasználói esetek diagrammal), a program szerkezetének leírását (UML osztálydiagrammal), valamint az esemény-eseménykezelő párosításokat és a tevékenység rövid leírását.

Feladatok

1. Potyogós amőba

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times m$ mezőből álló tábla (n az oszlopok, m a sorok száma), amelyre a játékosok X, illetve O jeleket potyogtatnak (azaz egy adott oszlopban a karakter mindig „leesik” a legalsó üres sorba, függetlenül attól, melyik sorban helyeztük le).

A játékosok felváltva lépnek, és egy oszlopban csak akkor helyezhetnek el új jelet, ha az még nem telt meg. A játékot az nyeri, aki előbb elhelyez vízszintesen, vagy átlósan négy szomszédos jelet. A játék döntetlennel ér véget, ha betelik a tábla. A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (8×5, 10×6, 12×7), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy melyik játékos győzött (ha nem lett döntetlen), majd kezdjen automatikusan új játékot.

2. Kitolás

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 játékos 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). 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, hogy melyik játékos győzött (ha nem lett döntetlen), majd kezdjen automatikusan új játékot.

3. Vadászat

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, majd kezdjen automatikusan új játékot.

4. Kiszúrós amőba

Készítsünk programot, amellyel a közismert amőba játék következő változatát játszhatjuk. Adott egy $n \times n$-es tábla, amelyen a két játékos felváltva X, illetve O jeleket helyez el. Csak olyan mezőre tehetünk jelet, amely még üres. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 5 egymással szomszédos jelet vízszintesen, függőlegesen vagy átlósan. A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjét kijelölve helyezhessük el a megfelelő jelet.

A kiszúrás a játékban az, hogy ha egy játékos eléri a 3 egymással szomszédos jelet, akkor a program automatikusan törli egy jelét egy véletlenszerűen kiválasztott pozícióról (nem biztos, hogy a hármasból), ha pedig 4 egymással szomszédos jelet ér el, akkor pedig kettőt.

A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (6×6, 10×10, 14×14), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy melyik játékos győzött (ha nem lett döntetlen), majd kezdjen automatikusan új játékot.

5. **Fekete lyuk**

Készítsünk programot, amellyel a közismert amőba játék következő változatát játszhatjuk. Adott egy $n \times n$-es tábla, amelyen a 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, majd automatikusan kezdjen új játékot.

6. 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.

7. 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.

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. 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, majd automatikusan kezdjen új játékot.

9. 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.

10. Rubik óra

Készítsünk programot, amellyel egy Rubik órát lehet kirakni.

A játékban 9 óra kap helyet, amely 1-12 közötti értéket mutatnak (kezdetben véletlenszerűen beállítva), és 3×3-as formában jelennek a játéktáblán. Az órák között az átlóknál 4 gomb helyezkedik el, amelyek 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 kezdő állásban az órák véletlenszerű időt mutatnak. 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.