5letek

+tanácsok

A projekt feladat sok meglepetést tud tartogatni. Tartsd szem előtt az alábbiakat!

A minőséget pontozzuk, nem a mennyiséget.

A program inkább legyen rövidebb, de jól megírt!

Olyan feladatot válassz, amely számodra izgalmas! Egy hozzád közel álló, neked érdekes feladaton sokkal hasznosabb és mellesleg könnyebb is dolgozni, még akkor is, ha programozási szempontból összetettebb. A túlvállalás, befejezetlen program viszont nagyon sokat ront a helyzeten!

Ne az utolsó pillanatban állj neki! A projekt feladat több követelménye strukturális jellegű, az egész programod felépítését érintő. Nagyon nehéz az utolsó pillanatban átdolgozni a teljes programot!

Számíts rá, hogy több idő lesz megírni a programot, mint amire először gondolnál. Az iparban minden ilyen jellegű időbecslést szinte gondolkodás nélkül megszoroznak kettővel. Sőt egy csomó dolgot itt nem rutinból fogsz csinálni, hanem ezen tanulod meg!

A hibakeresés hosszú időre meg tud akasztani. Könnyen lehet, hogy egy hibát csak akkor találsz meg, ha aludtál rá egyet.

A copy-paste nem javít, hanem ront a helyzeteden! Soha, de soha ne másolj kódot, inkább kérdezz! Később megtérül. Ha meg is írtál kódot másolgatva, hogy ki tudd próbálni, működik-e az ötleted, akkor is utána szüntesd meg a másolatokat. Kérj segítséget!

Ha lett egy >100 sorból álló függvényed, állj meg. Ne folytasd, semmiképp ne írd tovább... Törd darabokra, írj segédfüggvényeket, szervezz ki belőle részfeladatokat. Konzultálj, kérdezz az óraadótól, gyakorlatvezetőtől, társaidtól! Legjobb lenne, ha minden függvény ráférne egy képernyőre. (képernyőn itt egy 'oldschool' 80X25 -ös karakteres terminál értendő, nem egy 8K-s kijelző 6-os karakterrel...)

Beadás előtt ellenőrizd az összecsomagolt fájlokat, nem hiányos-e. Legjobb a ZIP fájlt ellenőrizni, tényleg benne van-e minden. Ellenőrizd a követelmények listáját is!

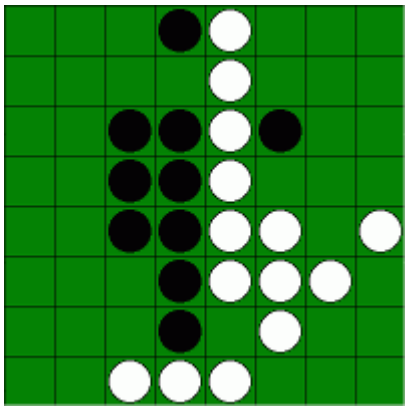
1. Játékok

Amőba *

Készíts menüvezérelt Python programot, mely amőbát játszik. A program legyen képes:

- a játék adminisztrálására egy tetszőleges méretű táblán,
- több támadó és védekező stratégiát alkalmazni,
- állás elmentésére és visszatöltésére.

Reversi **

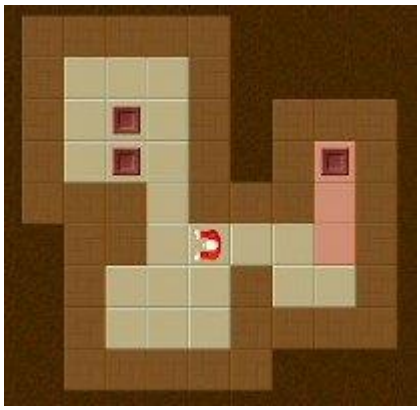


Írj reversi játékot, amelyben a gép ellen lehet játszani! A program legyen képes:

- Kirajzolni a pályát (szövegesen vagy grafikusan).
- Fájlba menteni és visszatölteni az állást.
- Ellenőrizni, hogy helyes-e a játékos lépése.
- Támadó stratégiákat alkalmazni.

(Az amőbával ellentétben ezt fix méretű pályán szokták játszani.)

Sokoban **



A játékban egy figurával ládákat kell tologatni a képernyőn; úgy, hogy azok a megfelelő helyre kerüljenek. A pálya viszont olyan, hogy könnyű betolni olyan helyre a ládákat, ahonnan már elmozdítani nem lehet őket.

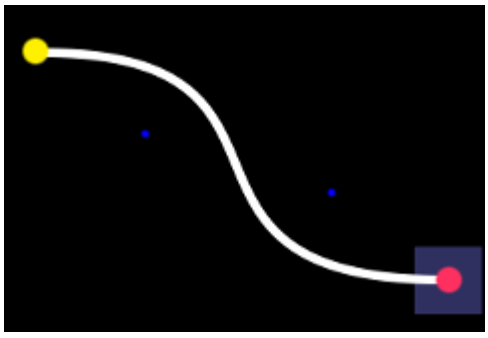
Olvasd a pályák leírását fájlból! Vezess dicsőséglistát! (Melyik játékos, melyik pályát, hány lépésből tudta megoldani?) Ez könnyedén megoldható szöveges képernyőn is.

Ehhez a játékhoz érdemes megcsinálni az XSB fájlok beolvasását. Egy ilyen fájl szövegként reprezentálja a pályát, pl. # a fal, szóköz a járat és így tovább. Ha ilyet tud a programod, rengeteg neten fellelhető pályával használható lesz, lásd például itt és itt.

Hexxagon **

Táblás játék. A pálya hatszögletű elemekből áll. Mindkét fél néhány bábuval indul. Minden lépésben a játékosok valamely bábuval a szomszédos helyre terjeszkedhetnek (ilyenkor nő a bábuk száma), vagy kettővel arrébb ugorhatnak (ilyenkor nem) egy szabadon választott bábujukkal. Az újonnan lerakott bábu mellett az ellenség bábui színt váltanak; a lépő játékos megnyeri azokat. YouTube videó itt: http://www.youtube.com/watch?v=_E10ydLaLE8.

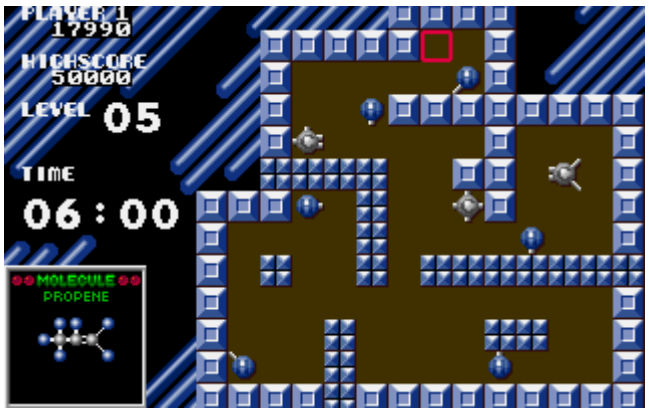
Írj programot, amelyben a gép különféle stratégiákkal játszik a felhasználó ellen!



Töltés-játék **

(Grafikus program.) A játék a következő. A program a képernyő adott pontjából, adott irányban kilő egy pozitív töltéssel rendelkező részecskét. A játékos számára adott néhány másik töltés, amelyeket úgy kell elhelyeznie a képernyőn, hogy a mozgó töltés egy megadott célba jusson. Ezek az elhelyezendő töltések lehetnek különböző erősségűek, polaritásúak. Közben a pályán lehetnek fix töltések, falak stb. (Hasonló játék itt: http://kmk.blog.hu/2007/07/26/newton_kedvenc_jateka)

A program számoljon pontszámot a játékos számára (pl. hány próbálkozásra sikerült megoldania a pályákat), és ez alapján tartson nyilván dicsőséglistát is, amelyet fájlba ment és vissza is olvas! A programnak tetszőlegesen sok töltést kell tudni kezelnie. A pályák leírását (hol van töltés, hol van fal) olvasd fájlból! Egy pályán lehessen tetszőlegesen sok mind a kettőből!

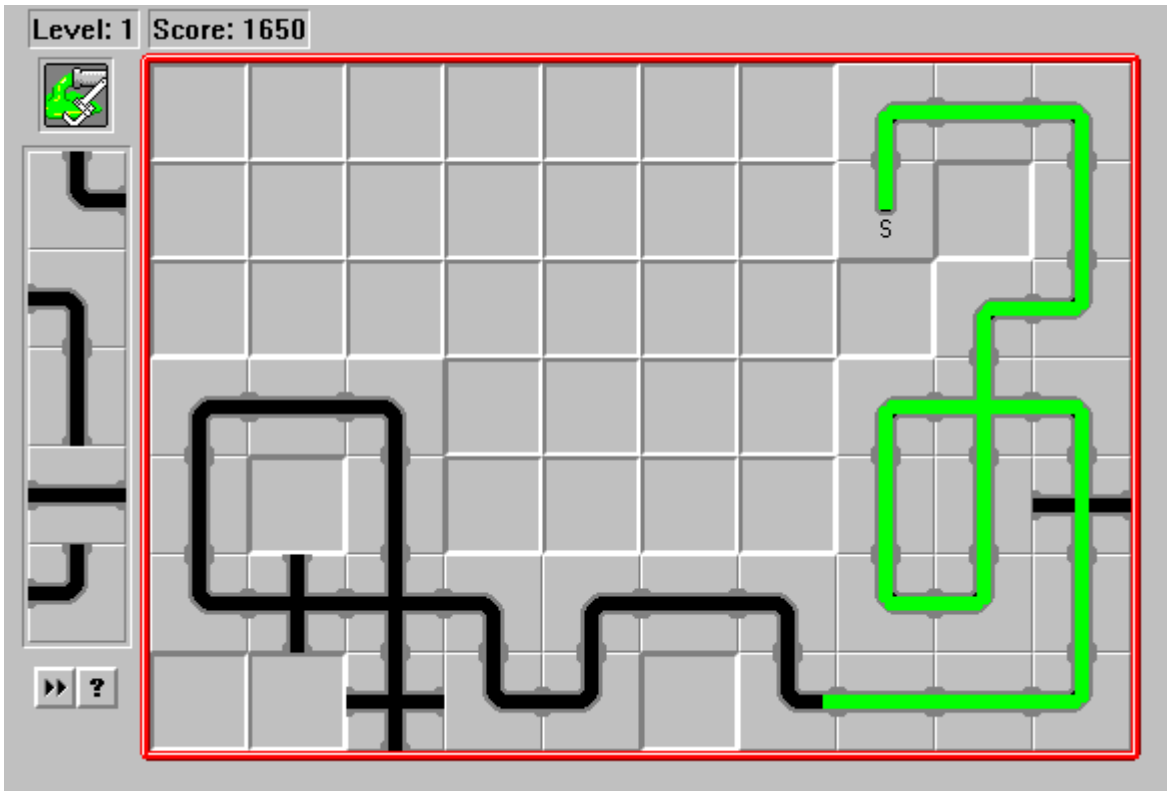


Laser Maze **

A Laser Maze egy gyerekeknek szóló fejlesztő játék, de létezik felnőtt változata is, nehéz feladványokkal.

A játék lényege, hogy egy adott pontból kiinduló lézerfényt kell tükrökkel egy megadott helyre irányítani. A kiindulás és a cél helye az adott pályától függ (honnan, milyen irányba induló fénysugarat kell eljuttatni hova, milyen irányból). A pályán lehetnek fix, nem mozgatható elemek is. A játékos az alábbi típusú elemeket használhatja fel, korlátos számban:

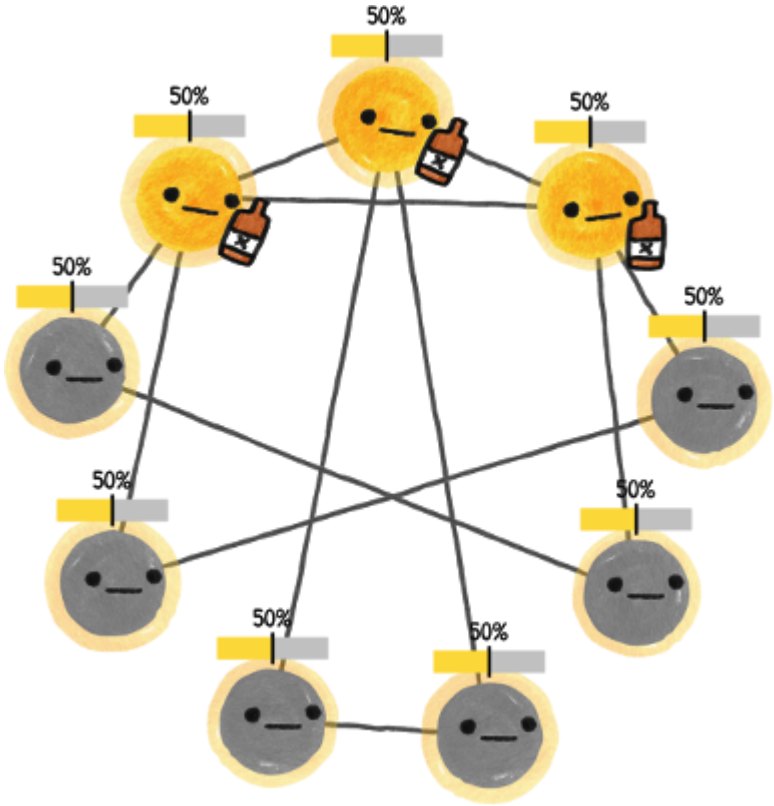
- Tükör (a rajzon ez kék).
- Féligáteresztő tükör, amely kettéosztja a fénysugarat (a rajzon zöld).
- Egyes pályán a fénysugarat blokkoló falak is lehetnek.



Pipe Dream **

A játékos véletlenszerűen csődarabokat kap (egyenesen menő, kanyarodó, egymást kereszteződő elemek). Ezekből kell egy hosszú vezetéket építenie a víznek, hogy eljuttassa a kiindulási helyről a célba. Többféle játék elképzelhető:

- Leghosszabb utat kell építeni.
- Legrövidebb utat.
- Időre megy – gyorsabban kell építeni a vezetéket, mint ahogyan a folyadék érkezik.



Közösségi hálók ***

Sok szociológiai jelenség megmagyarázható a kapcsolatok (kapcsolati hálók) ismeretével. Az alábbi ábra példát mutat erre. A jelenség itt: a közösség minden tagja úgy gondolja, hogy a közösség 50%-a alkoholista. Valójában ez nincs így, csak a közösség egyharmada nagyivó. Azért érzik így mégis, mert az ismerőseik 50%-a tényleg az, és ezt vetítik ki a teljes közösségre.

Írj programot, amelyben ez a jelenség modellezhető! A program jelenítse meg a közösség tagjait, és lehessen megadni a közöttük lévő kapcsolati hálót is. Ezek után a program számítsa ki, a közösség mely tagja mit gondol a teljes közösségről.

A rajz a The Wisdom and/or Madness of Crowds oldalról származik – ez további ötleteket is adhat a programhoz.

Farm **

A Farmville játékban egy veteményes kertet kell gondoznia a játékosnak: kapálnia, növényeket ültetnie, termést betakarítania. A feladat egy ilyen programot írni. A programnak tudnia kell:

- grafikusan megjeleníteni a kert állapotát (sok ingyenesen elérhető grafikai elem is van, pl. „plant tiles graphics” keresőszavak),
- megadható méretű kerttel dolgozni,
- az idő figyelembe vételével változtatni a kert állapotát,
- a játékállást fájlba menteni, visszatölteni, több játékállással dolgozni.

Kígyó játék **

Készíts kígyó játékot, ahol a kígyók ha megeszik a véletlenszerűen elhelyezett étket, akkor nő a méretük. Ha önmaguknak, vagy egymásnak ütköznek, az a játék végét jelenti. A program legyen képes:

- egy, illetve két játékos üzemmódra
- a pontok számolására és elmentésére

Laser Squad ***

Ebben a játékban két katonai csapat harcol egymás ellen. A játék körökre osztott: a csapatok katonái felváltva léphetnek, előbb egyik, majd a másik csapat. Minden katonának valahány időegysége van mozogni egy körben. Pl. lépésenként egy egységnyi időt használ fel, vagy az idejét lövésre is használhatja.

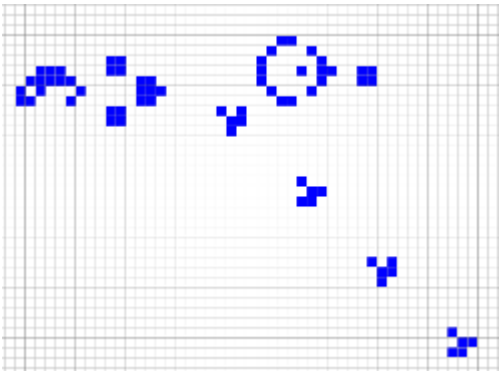
A játék szabályait tetszőlegesen specifikálhatod. Például lehetnek gyors (több időegységgel rendelkező) és lassú katonák. Lehetnek könnyen (kevés időegységért) és nehezen használható, de erősebb fegyverek stb. De ne vállald túl magad a specifikációban!

Aknakereső játék **

Készíts aknakereső játékot. A pálya téglalap alakú, cellákból álló tábla. Egy cellára rálépve megtudhatjuk, hogy az adott cellán van-e akna (ebben az esetben felrobbanunk – vége a játéknak), illetve, hogy hány szomszédos cellán van. Legyen lehetőség:

- a tábla méreteinek a meghatározására,
- a táblán lévő aknák számának megadására,

- a játékidő mérésére (esetleg limitálására),
- a játékos által aknának gondolt cellák megjelölésére,
- aknamentes környék automatikus felderítésére.



Életjáték **

Készíts menüvezérelt Python programot, mely a Conway féle LIFE (életjáték) modellt valósítja meg. Részletek a http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life címen. A program legyen képes:

- választott méretű élettéren,
- a felhasználó által,
- illetve fájlból beolvasott kezdeti állapotból indulni,
- a szimuláció folyamatos vagy lépésenkénti megjelenítésére.

Lift-szimulátor **

A program feladata liftek vezérlése.

- A liftekhez adott időközönként emberek érkeznek, akik választásod szerint jelzik úticéljukat (emelet száma), vagy úticéljuk irányát (fel/le).
- A programnak valamilyen algoritmussal irányítania kell a lifteket, kiszolgálni az utasokat.
- Az utasok a liftbe beszállva (ha az nincs túlterhelve) eljuthatnak a cél állomásra, ahol aztán kiszállnak. Amíg be nem fejezték az utazásukat, a programnak tételesen tudnia kell minden egyes utas adatait (hol van, hova készül stb.)

Bemenetként lehessen szövegfájlt megadni, amely a szimuláció adatait tartalmazza (pl. utasok megérkezésének ideje)! A program választásod szerint jelenítse meg az eredményt, vagy írja fájlba az utazások (kiszolgálások) adatait! A programod készítsen statisztikát az utazásokról (pl. várakozási idők, utazási idők)!

További játékok **/***

Készíts egyszerű játékprogramot (PacMan, Tetris, stb.) A program legyen képes:

- a játékot pontozással értékelni,
- a pontszámokat a játékos nevekkel együtt elmenteni (tetszőleges számban),
- az elmentett eredményeket a pontok szerint csökkenő sorrendben megjeleníteni,

- pálya térképet fájlból beolvasni (pacmannél), tetszőleges méretű pályán játszani (tetrisnél) stb.

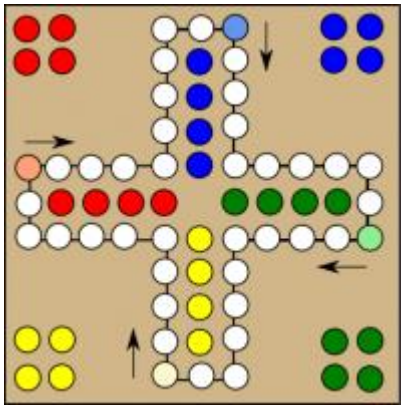
Scrabble **/***

A Scrabble nevű szókirakó játékban a játékosok betűket kapnak, amelyekből egy négyzetrácsos pályára értelmes szavakat kell kirakniuk, keresztrejtvényszerűen. Minél hosszabb szót raknak ki, minél több pályán lévő betűt felhasználva, annál több pontot kapnak (https://hu.wikipedia.org/wiki/Scrabble).

Készíts játékot, amelyben

- több játékos játszhat egymás ellen,
- a program a játékszabályokat ellenőrzi, a szavakat szótárból kikeresi,
- képes a játékállást fájlba menteni és visszatölteni,
- játékosok pontjai alapján dicsőséglistát készíteni.

A program sok irányba fejleszthető, sok nehezítés lehetséges. Pl. adhat a program tippeket a szótár alapján. Lehet tetszőleges méretű a pálya (előre megadott, vagy automatikusan növekedő). Lehet gépjátékos.



Ki nevet a végén ***

A klasszikus Ki nevet a végén? társasjáték gépesített változata. A játékosok egymás ellen játszanak, a gép a kockadobásokat szimulálja és a játékszabályokat ellenőrzi.

Legyen képes a program:

- a pályát megjeleníteni,
- a játékosok lépését kezelni és ellenőrizni,
- a játékállást fájlba menteni.

A program megvalósítható grafikusan és konzolon is.

Kártyajáték **/***

Készíts programot, amelyik valamilyen kártyajátékot valósít meg: römi, póker stb. A programnak nem kell tudnia játszani a felhasználó ellen, viszont:

- Kezelje a lapokat listában.
- Jelenítse meg az állást.
- Legyen lehetőség az állás mentésére és visszatöltésére.

Megvalósítható szöveges és grafikus képernyőn is.

Legyen Ön Is Milliomos! *

Mint a tévés játék: egymás utáni kérdések, mindegyikre egy jó és három rossz válasz. Rossz válasz esetén rögtön kiesik a játékos; lehet kérni a gép segítségét (elvesz két rossz választ) és a közönség segítségét (szavaznak). A program tartalmazzon dicsőséglistát a játékosokról: ki mennyi pénzt nyert, és mennyi ideig (perc) volt játékban.

A programnak fájlból kell beolvasnia a kérdéseket, amelyekből tetszőlegesen sok lehet. A táblázat minden kérdéshez egy nehézségi szintet is meghatároz; a létrehozott adatszerkezetnek a kérdéseket nehézség szerint kell csoportosítania, és a játékosnak választania kell tudni induláskor egy nehézségi szintet.

A program alkalmazzon grafikus, vagy legalább színes konzolos megjelenítést!

2. Matematikai jellegű programok

Numerikus integráló ***

Készíts Python programot, mely a felhasználó által megadott függvényt numerikusan integrálja. Gondosan tervezd meg az adatstruktúrát! Tervezz egy megfelelő bemeneti nyelvet. A program legyen képes:

- alapműveletek, polinomok,
- a szokásos matematikai függvények (sin, log, exp stb.)
- és tetszőleges kombinációjuk kezelésére.

A program legyen képes felhasználó által megadott kifejezéseket eltárolni és feldolgozni! (Ezek lehetnek fordított lengyel jelöléssel adottak is.)

Mátrix függvénykönyvtár **

Készíts mátrixszámításokhoz függvénykönyvtárat a Python kiegészítő függvénykönyvtárainak (pl. numpy) használata nélkül. Tárold egy mátrix szélességét, magasságát, valós értékeit! Legyen képes a programod bármilyen nagy mátrixokkal dolgozni, és a szokásos műveletek elvégzésén kívül a mátrixokat fájlba írni és fájlból visszaolvasni!

Valósíts meg a mátrixokon gyakran használt algoritmusokat: Gauss-elimináció, inverz mátrix, mátrix rangja, determináns stb.

Gráf tárolása ***

Készíts programot, amelyben gráfot tárolsz! Lehessen a gráfnak tetszőlegesen sok csúcsa, éle; lehessen csúcsot, élt hozzáadni és törölni! Lehessen a gráfot fájlba menteni, fájlból beolvasni!

A gráf alkalmazására tetszőleges feladatot kitalálhatsz. Pl. tárolhatod a csúcsok koordinátáit, hogy ki tudd rajzolni az, vagy tárolhatod az élek súlyát (távolságot), hogy legrövidebb utat keress Dijkstra algoritmusával. Megvalósíthatsz szélességi és mélységi bejárást.

Négyszín-tétel ***

A négyszín-tétel azt állítja, hogy bármely térkép országai kiszínezhetők négy különböző színnel. Formálisan: ha egy síkot tetszőlegesen összefüggő részekre osztunk, akkor ezek mindegyikéhez hozzárendelhetjük a négy szín egyikét úgy, hogy semelyik szomszédos rész nem lesz egyforma színű.

A feladat egy játékot írni, amely erre épít. A program véletlenszerűen felosztja a játékteret síkidomokra, a felhasználónak pedig ki kell színeznie azt. (Egy egyszerűbb változat: nem síkidomokat, hanem gráfokat ad a program, amelynek szomszédos csúcsait kell színezni négy különböző színűre. Vigyázat: ez nem lehet akármilyen gráf.)

Ötletek:

- Mért játékidő, pontozás.
- Játékállás mentése, betöltése.
- Dicsőséglista.
- Véletlenszerű térképek/gráfok újragenerálása.

Gráf síkbarajzolása ***

Némely gráfok síkbarajzolhatóak, azaz úgy, hogy az éleik nem keresztezik egymást. Az ilyen gráfok csúcsai úgy is elhelyezhetőek, hogy az éleket egyenes vonalakkal (szakaszokkal) sem fogja semelyik él keresztezni a másikat.

A feladat egy játékot írni. A program véletlenszerű gráfokat rajzol ki, amelyeknek csúcsait a játékos kell úgy elhelyezze, hogy sehol ne legyenek egymást keresztező élek.

Ötletek:

- Mért játékidő, pontozás.
- Játékállás mentése, betöltése.
- Dicsőséglista.
- Véletlenszerű térképek/gráfok újragenerálása.
- A program generál nem síkbarajzolható gráfot, és a játékos meg kell mondja, hogy nem az,
- Vagy a program nem generál ilyet (de ekkor ellenőrizni kell a véletlenszerűen generált gráfokat).

Útvonaltervező *

Készíts útvonaltervező programot! A program legyen képes egy térkép szöveges reprezentációját fájlból beolvasni és az ebből felépített adatstruktúra alapján két megadott helyszín között útvonalat tervezni. Lehessen megadni az útvonaltervezés szempontját is (leggyorsabb, legrövidebb, stb)!

Buszjáratok *

Készíts programot, amely egy közlekedési társaság buszjáratait képes nyilvántartani, és a menetrendek alapján útvonalakat felépíteni! Elvárások a programmal szemben:

- tudja tárolni megállók neveit
- tárolja járatok adatait, amiben a fenti megállónevek szerepelnek,
- legyen képes útvonalakat megtalálni egy adott helytől egy másik helyig (két megállónévvel adott), átszállásokkal együtt
- tárolja a járatok menetrendjeit is (melyik órában hány percenként)
- ezek alapján számolja ki egy utazás minimum és maximum időtartamát is
- minden adatot mentsen fájlba, hogy azokat később vissza lehessen tölteni és tovább szerkeszteni

Koordinátageometria **

Írj a Geogebra nevű programhoz hasonló programot! Legyen képes a program köröket, szakaszokat, egyeneseket tárolni a síkon, és egyszerűbb szerkesztési lépéseket elvégezni! Például metszéspontok meghatározása, adott pontban merőleges állítása stb.

A programnak tetszőlegesen sok alakzatot kell tudnia tárolnia, és egy rajz alakzatait (körök, szakaszok, egyenesek) fájlba menteni, onnan betölteni.

Javasolt megjelenítési módszert keresni: pl. SDL-es grafikával, vagy SVG fájlba írással.

Koordinátageometria++ ****

A feladat ugyanolyan, mint az előzőnél, de építs be grafikus megjelenítést, és tedd lehetővé a felhasználó számára, hogy egérrel adja meg az alakzatokat!

3. „Számítástechnika, algoritmusok” feladatok

Rendezőalgoritmusok **

Készíts függvénykönyvtárat, amely különféle rendezéseket valósít meg listákra! Demonstráld, teszteld a függvénykönyvtár működését különféle listákon! A programnak kötelezően:

- Tartalmaznia kell olyan rendezéseket (többesszám!), amik az erről szóló előadáson nem szerepeltek.

Készíts egy alkalmazást a függvénykönyvtárhoz! Például egy olyat, amely beolvas egy szövegfájlt, és az abban soronként tárolt neveket ábécé rendbe rendezi, majd kiírja egy másik fájlba.

Képfeldolgozás ***

Készíts programot, amelyik képes képfájlok betöltésére, és mentésére, továbbá azon egyszerű képfeldolgozási lépések végrehajtására!

- A képek betöltését és mentését neked kell megvalósítanod.
- Hajts végre különböző műveleteket a képen, pl. sötétítés, világosítás, kontrasztnövelés, elmosás, élkeresés stb. (Ezeknek érdemes utánaolvasni, esetleg rákeresve a „konvolúciós mátrix” fogalomra. Sok ilyen művelet mögött meglepően egyszerű a matematika.)
- Grafikus megjelenítés opcionális, a program működhet parancssori alkalmazásként is.

Több olyan képfájl formátum van, amely tömörítetlen adatot tartalmaz, és a feldolgozása a tananyag alapján elvégezhető. Ilyen például a Netpbm is (

http://netpbm.sourceforge.net/doc/ppm.html). Dolgozz ilyennel!

Archív fájl függvénykönyvtár ***

Készíts függvénykönyvtárat, amellyel egy archív fájlba becsomagolt fájlok nyithatók meg Python programból! A függvénykönyvtár használata hasonlítson minél jobban a Python fájlkezelésére!

A fájlok becsomagolását egy parancssori segédprogrammal lehessen végezni:

```
pack archiv.dat file1.bmp file2.dat file3.doc
```

Ekkor keletkezzen egy **`archiv.dat`** nevű fájl. Abból programból a **`file2.dat`** megnyitása:

```
file = archive_open("archiv.dat", "file2.dat", "r");
buffer = file.read(1024);
```

A fájlokat tömöríteni nem kell.

Parancssor program ****

Írj parancssor (shell) programot! A program legyen képes arra, hogy

- a begépelt nevű másik programokat elindítsa,
- a szabványos bemenet, szabványos kimenet átirányítására,
- csővezeték (pipe) létrehozására,
- változók létrehozására, értékük behelyettesítésére,
- parancsnév-rövidítések (aliasok) létrehozására.

Figyelem: ezt a feladatot kizárólag olyanoknak ajánljuk, akik Unix (Linux) rendszereket valamennyire ismerik, és ilyenen szeretnék megvalósítani a programot.

Huffman kódoló ****

Készíts parancssorból hívható Python programot, amely a Huffman kódoló algoritmust felhasználva tömörít fájlokat. Részleteket lásd a http://en.wikipedia.org/wiki/Huffman_code címen. Tervezz megfelelő fájl formátumot a tömörítés kódtáblázatának tárolására. A program legyen képes:

- fájlok tömörítése,
- fájlok visszaállítása,
- a tömörítés iránya parancssor kapcsolóval legyen megadható.

Lempel-Ziv tömörítő ****

Lempel és Ziv algoritmusa a következő ötlettel tömöríti a fájlokat. Tegyük fel, hogy adott egy fájl a következő tartalommal:

```
Blah blah blah
```

Ebben a színnel jelölt részek egyformák, ezért a második előfordulást egy hivatkozással lehet helyettesíteni:

```
Blah b[D=5,L=5]lah
```

Ez azt jelenti, hogy D=5 bájtnyival ezelőtt volt egy L=5 hosszúságú sorozat, amit az adott helyen meg kell ismételni. (Most hosszabbnak tűnik, de binárisan tárolva ezt, rövidebb lesz.)

Írj programot, amely ilyen módszerrel képes tömöríteni és kicsomagolni fájlokat! Teszteld a programodat szövegfájlokon és képeken is (pl. BMP. A .png és .jpg fájlok már tömörítve vannak, azokat hiába próbálod tovább préselni.) Az algoritmusról sok írást találsz a neten.

Reguláris kifejezések ***

Járj utána, mik azok a reguláris kifejezések (regular expression)! Ezekről van szó előadáson, de a feladat megoldásához a tárgyon túlmutató tudás szükséges.

A reguláris kifejezések állapotgéppel feldolgozhatóak: https://www.youtube.com/watch?v=GwsU2LPs85U. Írj programot, amelyik ilyen átalakítást végez! Tehát egy megadott reguláris kifejezésből felépít egy állapotgépet, amelyikkel aztán meg tudja mondani egy sztringről, hogy illeszkedik-e rá.

A felépített állapotgépet természetesen a programban reprezentálnod kell valahogy – oldd meg ezt két dimenziós állapottáblával vagy állapotátmeneti gráffal, amit felépítesz adatszerkezetként!

Készíts egyszerű szűrőprogramot, amely a parancssori **`grep`** programhoz hasonlóan egy szövegfájlból kiválogat olyan sorokat, amelyek illeszkednek egy megadott reguláris kifejezésre.

4. Nyilvántartás jellegű programok

Ezeknél a programoknál mindig van valami, ami egyedivé teszi azt a többi feladathoz képest. Az egyedivé tevő funkcionalitás (pl. telefonkönyv esetén vCard export/import, étterem esetén alaprajz stb.) nem hagyható el a programból.

Telefonkönyv *

Készíts menüvezérelt Python programot, amely „rekordokban” tárolja bizonyos személyek nevét, foglalkozását, címét, esetleg más jellemző adatait. A program legyen alkalmas:

- új rekordok létrehozására, a régiek módosítására, törlésére,
- név, telefonszám, egyéb mezők szerinti keresésre,
- név esetén egy darab *-ot tartalmazó helyettesítés (wildcard) kezelésére (pl. a "Nagy*", "N*án", "*István" keresősztringek mindegyike megtalálja Nagy Istvánt a telefonkönyvben),
- az adatbázis fájlba mentésére,
- egy bejegyzés vCard formátumba történő exportálásra, vCard fájl beolvasásra (azokat a mezőtípusokat, amiket a programod ismer).

Teszteld a vCard fájl írását, beolvasását a telefonodról mentett, és a programodból generált, a telefonodra elküldött fájllal!

Határidőnapló *

Készíts határidőnapló programot, amely „rekordokban” tárolja az események dátumát, pontos idejét, helyét, elnevezését, és egy hozzá kapcsolódó megjegyezést. A program legyen képes:

- új rekodok létrehozására,
- a régiek módosítására,
- a régiek törlésére,
- egy adott naphoz, héthez és hónaphoz tartozó események naptárszerű (időrendes) kilistázására,
- esemény név szerinti keresésére,
- az adatbázis fájlba mentésére, visszatöltésére.

Étterem *

Készíts programot, amely egy étteremben az egyes asztalokhoz tartozó megrendeléseket jegyzi meg. Legyen lehetőség:

- Az asztalok megadására: hány fős, hol helyezkedik el az éttermen belül.
- Az étterem menüjének rögzítésére.
- Új asztal nyitására vendégek érkezése esetén.
- Rendelések felvételére a menü alapján.
- Számla „nyomtatására” (képernyőre).

- Foglaltsági térkép megjelenítésére, figyelembe véve az asztalok elhelyezkedését (grafikus vagy egyszerű konzolos felületen).

Az adatokat mentse a program fájlba is, hiszen azokat nem felejtheti el egy újraindítás miatt!

Mini-Facebook *

Írj programot, amely emberek személyes adatait (név, nem, születési dátum stb.), továbbá ismeretségi viszonyait (kik ismerik egymást) képes nyilvántartani, és lehetővé teszi azt, hogy két ismerős üzenni tudjon egymásnak!

A programnak nem kell grafikusnak, se hálózatosnak lennie; elég, ha egyszerre egy ember tudja használni. (Utána ő kijelentkezik, és valaki más bejelentkezik.) Tegyen lehetővé ugyanakkor kereséseket (név szerint, iskola szerint, lakóhely szerint stb.), és ajánlja fel egy menüpontban a felhasználók számára az ismerősök ismerőseit! Tároljon el minden adatot fájlban (az üzeneteket is), és tegye lehetővé akárhány felhasználó és üzenet létezését!

Könyvtár *

Készíts könyvtár-programot, amely képes könyvek adatait eltárolni. A program legyen képes:

- új könyvek létrehozására, a régiek módosítására, törlésére,
- a könyvek szerző, cím, kiadási év, téma alapján való keresésére.

Legyen képes ezen felül:

- Olvasók adatainak kezelésére,
- a kölcsönzés adatainak kezelésére (melyik könyv épp kinél van),
- megmondani egy adott könyvről, hogy az ki van-e kölcsönözve, és ha igen, kinél,
- kilistázni egy adott olvasónál lévő könyveket.

Mentsd az összes adatot fájlba, hogy újraindítás esetén se vesszenek el az adatok!

Repülőjegyek *

Készíts menüvezérelt Python programot, amellyel repülőjegyek foglalásait tudod nyilvántartani! Legyen lehetőség a programban a járatok adatait megadni. Ezen felül legyen lehetőség *külön menüpontban* elvégezni olyan műveleteket, amelyeket általában internetes repülőjegy-vásárlásnál is lehet:

- Járatokat keresni.
- Adott névre repülőjegyet foglalni.
- Egy megkeresett foglaláshoz ülőhelyet rendelni.
- Egy megkeresett foglaláshoz menüt (normál, vega, laktózmentes stb.) választani.

Legyen képes a program:

- Kezelni az ülőhelyek számát, nehogy túlfoglalás legyen
- Ne engedje két utasnak ugyanazt az ülőhelyet kiadni

- Az ülőhelyekről foglaltsági térképet megjeleníteni (választás szerint konzolos vagy grafikus felületen)
- Összesíteni, hogy melyik járatra, melyik menüből hány adagot kell felvinni.