Feladatok

11 Fáljok és elérés utak

1. Írja ki a program aktuális könyvtárában lévő összes könyvtár és fájl nevét. Egyszer a könyvtrárakat, utána a fájlokat.

2. Írja ki a program prancssori argumentumaként megkapott könyvtárban lévő összes könyvtár és fájl nevét. Egyszer a könyvtrárakat, utána a fájlokat.

3. Írjunk egy programot ami az aktuális könyvtárban létrehoz egy _DataFiles könyvtárat, majd a consoleról *-ig beolvasott nevű fájlokat hoz létre ebben a könyvtárban.
   (Próbáld ki, hogy mi történik, ha olyan fájlneveket adsz meg, amik nem kanonikus utak. Mit gondolsz, miért lehet ebből baj?)

4. Írj egy programot ami addig kér be könyvtárneveket, ameddig abban pont nem lesz (pl x, y, a.txt), majd ez alapján létrehozza az aktuális könyvtárban az x/y/a.txt fájlt.

5. Írjon egy programot, ami a parancssori argumentumaként kapott fájlból feltölt egy egész számokból álló tömböt. A fájl első sorában a feltöltendő tömb hossza van. Utána egy üress sor. Majd a tömb elemei szóközzel elválasztva.

6. Írjon egy programot, ami a parancssori argumentumaként kapott fájlból feltölt egy karakterláncokból álló tömböt. A fájl első sorában a feltöltendő tömb hossza van. Utána egy üress sor. Majd a tömb elemei soronként.

7. Írjon egy programot, ami a parancssori argumentumaként kapott fájlból feltölt egy kétdimenziós törtszám tömböt, aminek a második dimenziói változnak. A fájl első sora azt adja meg, hogy a tömb első dimenziója mentén hány elem van. Ez után minden sor egy számmal kezdődik, ami azt mutatja meg, hogy a második dimenzióban a tömb hány elemet tartalmaz abban az elemben. Majd ezek a számok következnek szóközökkel elválasztva.
   Pl.:

   ```
   2
   5 1 2 3 4 5
   3 1 2 3
   ```

   azzal egyenértékű, hogy {{1, 2,3, 4 ,5}, {1, 2, 3}}.

8. Írjon egy programot, ami a parancssori argumentumaként kapott fájlba kiír egy 100x100-as tömböt
   a. az előző feladat formátumában szöveges fájlba és
   b. bináris fájlba, csak a számokat kiírva
   c. majd írja meg a bináris formátumból való visszaovlasást is.

d. Ezen kívül írjon egy ellenörző függvényt, ami a bináris fájlból illetve a szövegesből beolvasott tömböket összehasonlítja.

9. Írjon programot, ami a prancssori argumentumként kapott fájlból felolvas egy labirintust. A labirintus egy kétdimenziós karaktertömbben tárolható. A fájl első sorában két szám szerepel szóközzel elválasztva, ami a labirintus méretét adja meg (sorok és oszlopok száma). Ez után a sorokban x jelöli a falat és szóköz a szabad járható részeket.
(Értelmes labirintust generálni nehezebb mint felolvasni, vagy akár kitalálni belőle, azt majd egy hónap mulva.)

12 Állapotgépek

1. Írjon egy programot, ami megvalósítja egy forgóvilla állapotgépét (nyitott és zárt állapot, pénzbetételre egy átforgást engedélyez, amúgy nyomásnak ellenáll).
Egy 1000 hosszú optimális eseménysorozat lenne, hogy 500x ismétlődne a pénzbedobás, átmenés sorozata. Próbálja ki, hogy hány sikeres átjutás történik, ha az 1000 próbálkozás mindegyike véletlenszerűen pénzbedobás vagy átmenési kisérlet.

2. Írjon egy állapotgépet ami egy fotocellás ajtó működését modellezi. Modellezze, hogy milyen állapotai lehetnek, illetve milyen ingerek érhetnek egy fotocellás ajtót.

3. Egy karakterláncon karakterenként végig haladva (mintha egy karakterfolyam lenne, és nem állna rendelkezésre az egész) dolgozza azt fel. A feldolgozás eredménye legyen egy karakterlánc tömb, amibe a bemeneti karakterlánc a parancssori argumentumok darabolási szabályai szerint van felvágva (szóközök mentén darabolva, de ”-ek közötti részeket egyben tartva).
Pl.: Alma elment aludni mert “nagyon fáradt” volt -> {“Alma”, “elment”, “aludni”, “mert”, “nagyon fáradt”, “volt”}.

13 Rendezések

1. Írjon egy programot ami az aktuális könyvtárban lévő bemenet.txt fájlból felölt egy karakterlánc tömböt. A fájl első sorában a későbbi sorok száma van, utána soronként egy-egy karakterlánc.
   a. Rendezze a tömböt buborék rendezéssel.
   b. Rendezze a tömböt kiválasztásos rendezéssel.
   c. Rendezze a tömböt beszúrásos rendezéssel.
2. Írjon egy programo ami a 14.9 feladat rendezetlen adatbázisából készít egy rendezett verziót.

14 Rekurzió

1. Írjon rekurzív algotmut, ami egy tört számot megszoroz egy egész számmal, kizárólag összeadás művelet használatával.

2. Írjon ki a képernyőre n darab csillagot. Oldja meg kétféleképpen is. Iteratív módon és rekurzió felhasználásával.

3. Egy Hanoi-i templomban a legenda szerint van 3 oszlop, amiből az elsőre lyukas korongok voltak felfűzve. Alul a legnagyobb, majd csökkenő sorrendben. Ezeket szerzetesek mozgatják olyan szabályok szerint, hogy
    a. egyszerre csak egy korongot lehet mozgatni,
    b. a mozgatott korong csak üres oszlopra, vagy egy nála nagyobb korongra tehető.

   A legenda úgy szól, hogy amikor az összes korongot átmozgatnák a harmadik oszlopra, akkor eljő a világvége. Segítsünk nekik.
   Írjon egy programot ami vizuálisan megmutatja a Hanoi tornyai feladat megoldását n számú korong esetén (4,5 körül próbálkozzunk).
4. Egy n egész szám esetén $n!$ jelöli 1-től n-ig a számok szorzatát. $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$. Írjon rekurzív algoritmust amely kiszámolja $n!$ értékét.

5. A Fibonacci sorozat elég természetellenes képességű nyulak szaporodását modellezi. A sorozat úgy indul, hogy $F_2 = F_1 = 1$. Ezek után egy tetszőleges F_n érteke úgy számolható ki, hogy a sorozat két előző elemét összedadjuk $F_n = F_{n-1} + F_{n-2}$. $(1, 1, 2, 3, 5, 8, 13, \ldots)$
   Számolja ki F_n értékét rekurzió segítségéve.

6. Írjon egy programot ami két parancssori argumentumot vár. Az első egy könyvtár elérési útja, a második egy fájl neve. Egy rekurzív függvény segítségével döntse el, hogy van-e olyan fájl a könyvtárban (vagy bármely abban lévő könyvtárban tetszőleges mélységben) bárhol olyan nevű fájl. Ha vannak ilyenek, ezeknek a teljes elérési útját írja ki a consolere.

7. Egy olyan világban mozgunk, ahol a koordináták egész számok. És utazni csak egy fura szabály szerint lehet. Egy (x, y) koordinátájú pontból csak az $(x, x + y)$ és az $(x + y, x)$ koordinátákra lehet mozogni.
   Írjon egy függvényt, ami eldönti, egy indulási és egy érkezési koordinátapárról, hogy az érkezési pont elérhető-e az indulási helyről.
   Pl.: indulási hely $(2{,}10)$, érkezési hely $(26{,}12)$ esetén $(2, 10) \rightarrow 2, 12) \rightarrow (14, 12) \rightarrow (26, 12)$ egy megengedett útvonal. Indulási hely $(2{,}10)$, érkezési hely $(15{,}16)$ esetén nincs ilyen útvonal.
   (Bónusz: írja is ki a lépések sorozatát. Visszafelé egyszerűbb, előre kicsit bonyolultabb.)

8. XML és egyéb rekurzív fájlformátumok fontos szerepet játszanak a szoftveriparban. Ezeknek a kezelésére rekurzív feldolgozónyelveket használunk. XML fájlok feldolgozására, azok tartalmának címzésére az XPath kifejezéseket használjuk. Ennek egy alap formáját készítitek el ebben a példában.
   Egy xml struktúra tagekből áll, melykenk elejét a <tagnév> karaktersorozat jelöli és a </tagnév> karaktersorozat zárja. Egy tag címzésére egy olyan karakterlánc használható ami tagnév1[hanyadik1]/tagnév2[hanyadik2]/…/ tagnévN[hanyadikN] alakú és a zárójelben lévő számlálók azt mutatják, hogy a tag a vele egy szülő alatt lévő, azonos nevű tagek közül hanyadik.
   Írj programot ami két parancssori argumentumot vár. Az első a feldolgozandó xml fájl elérési útja. A második pedig egy fenti formának megfelelő kifejezés. Ezek alapján kiírja a fájlban szereplő összes szöveget (ami nem tag markup) és közvetlenül a megcímzett tag alatt van (tehát nem mélyebb tagek alatt).

(Ebben a feladatban fájlműveleteket, karakter és karakterlánc műveleteket, állapotgépet és rekurziót is használunk.)

9. Egy irodaházpark irodái egy olyan címmel azonosítódnak, amiben az épület azonsóítója (karakterlánc), azon belül az emelet (szám) és azon belül az iroda száma (szám) van pontokkal elválasztva. Az irodák adatbázisa egy fájl, aminek első sora tartalmazza az adatbázisban lévő sorok számát, utána pedig soronként tartalmaz egy címet és az ott székelő cég nevét. Pl.: Neumann.12.321. ACME
   Írjon egy programot, ami egy felhasználó által megadott irodacímet megkeres az adatbázisban és kiírja a megfelelő irodához tartozó cég nevét.
   a. A feladatot oldja meg a rendezettlen adatbázisra
      i. rekurzívan és
      ii. iteratívan.
   b. A feladatot oldja meg a rendezett adatbázisra.
      i. rekurzívan és
      ii. iteratívan.
   c. Mérje le az előző megoldások futási idejét és vonja le a megfelelő tanulságokat.

   (Ebben a feladatban nem csak rekurziót, de a keresési algoritmusokat is használjuk.)

15 Backtracking

1. Egy $n \times n$ -es méretű saktábblán írja ki a képernyőre az összes olyan elrendezést, ahol pontosan n királynő van és azok nem támadják egymást a sakk szabályai értelmében.

2. Kosárcsapatokat szeretnénk összekombinálni adott pénzkeretből. A bemeneti adatok egy fájlban vannak, amit parancssorról kapunk meg. A fájl első sorában a rendelkezésre álló pénzösszeg áll, a második sorában az áll, hogy hány játékos van a piacon. Ez után soronként minden játékos ára van megadva, majd utána a neve.
   Írjunk egy olyan programot ami kiírja az összes olyan csapatkombinációt, amiben van legalább 5 játékos és bele is fér a megvételük a rendelkezésre álló keretbe. (A,B,C és A,C,B névösszetétel azonosnak számít, az ilyen csapatokat csak egyszer írjuk ki.)

3. Írjuk ki egy n pozitív egész szám összes összegfelbontását, vagyis az osszes olyan felbontást, amiben pozitív egész számok összegeként előállítható.
   Pl.: $4= 1+1+1+1 = 1+1+2 = 1+3 = 2+2$
   (a 3+1, 1+2+1 és hasnoló kombinációkat ne írjuk ki)

4. Egy zászlókészítő mogyoró, fenyő és akác és bükk fából készít zászlórúdakat, amire vászon, selyem és műszálas drapériát tud akasztani. A drapériára vászon és selyem esetén csak selyem bolytot lehet rakni, a műszálas drapéria csak műszálas bolyttal kapható. A zászlók tetjére zászlócsúcsdísz is van, de csak selyem és vászon drapéria esetén. Selyem drapériára ezüsttel vagy arannyla futtatott csúcsdísz, illetve réz is kapható, a vászonhoz csak réz.
   Backtracking használatával írja ki az összes zászlókombinációt, ami elrhető a kínálatban.

5. (15. 2. Folytatás) Keressük meg azt a csapatot, aminek összköltsége a legközelebb van a rendelkezésre álló kerethez.
   (Ne tároljuk el az összes lehettséges megoldáscsapatot, csak az aktális legjobbat frissítsük a backtrackingben és a végén írjuk ki egyszer.)

6. Egy parancssori argumentumként bemeneti szöveges fájlban a napra tervezett megbeszélések kezdeti és végideje van megadva (az egyszerűség kedvéért tört órában, pl. 13.5 jelenti a fél kettőt). A fájl első sorában a tervezett megbeszélések száma van. Utána minden sorban egy-egy megbeszélés kezdeti és vég időpontja.
   Keresse meg az összes olyan megbeszélés-kombinációt, ami ütközés nélkül ütemezhető. A keresés során válassza ki azt a kombinációt, amelyben a legtöbb megbeszélés kerül ütemezésre, illetve azt is, ami a nap legnagyobb részében kihasználja a tárgyalót.

7. Tekintsük az alábbi mondatot:
   Ebben a mondatban az
   - 1-es számjegy ___ ,
   - a 2-es számjegy ___,
   - a 3-as számjegy ___,
   - a 4-es számjegy ___,
   - az 5-ős számjegy ___,
   - a 6-os számjegy ___,
   - a 7-es számjegy ___,
   - a 8-as számjegy ___ és
   - a 9-es számjegy ___ esetben jelenik meg.

   Írjon egy programot, ami úgy adja meg az üres helyekre írandó számokat, hogy a mondat által megfogalmazott állítás igaz legyen. Felteheti, hogy egyik szám szerepel többször, mint 20.

8. Írjunk egy egyszerű Sudoku megoldó algoritmust. A bemeneti fájlformátum 9 sort tartalmazoon, amelyik mindegyike 9 karaktert tartalmaz. Az ismert helyeken a számok, az ismeretlenek helyén * legyen. Írja ki az első megoldását a fájlban található Sudoku feladványnak.

9. Egy $n \times n$ -es négyzetrács csúcspontjaiban szeretnénk minél több pontot elhelyezni úgy, hogy azok között semelyik három ne legyen egy vonalon. Írjon programot, ami adott n érték esetén meghatározza a legtöbb elhelyezhet pont számát.
   (Érdekesség: $n < 46$ esetén tudjuk, hogy $2n$ pont elhelyezhető, a tudomány jelenlegi állása szerint az a sejtés, hogy nagyobb n értékekre szigorúan csakis kevesebb mint $2n$ pont helyezhető el. A megoldási lehetőségek száma annyira magas, hogy ennek a kérdésnek az eldöntését nem a backtracking fogja meghozni, kis n értékekkel teszteld az algoritmust.)

10. Egy osztály általános iskolás most végzett és középiskolába mennek. Minden diák megjelöl 3 barátot. A diákok elosztási folyamatának azt kell garantálnia, hogy minden diák úgy kerül beosztásra, hogy legalább egy barátja is vele egy osztályban van. Egy diák azt hallotta 5 másik diáktól, hogy ők kitalálták, hogyan adhatnák meg a 3 barátjukat, hogy azzal mind az 5-en ugyanabba az osztályba kerüljenek.

Írj egy programot, ami n diák esetén eldönti, hogy van-e ilyen stratégia és ha van, meg is adja, hogy melyik diák melyik 3 barátot kell megjelölje. Ennek segítségével találd meg a legnagyobb olyan csoportméretet, aminél még garantálható, hogy a csoport együtt marad a diákok szétosztása során.

Valóban létezik olyan stratégia 5 diák számára, ami garantálja, hogy egy osztályba kerüljenek?