INFORMATIKA – tétel
2023

ELIGAZÍTÁS:

- 1 pont hivatalból;
- Az 1-4 feladatokban (a pszeudokód programrészletekben): (1) a **`kiír '\n'`** utasítás új sorba ugratja a képernyőn a kurzort, **`kiír ' '`** szóközt ír; (2) a „**/**” operátor osztási hányadost ad meg, a „**%**” operátor pedig osztási maradékot.
- Az 5-9 feladatok esetében használj alprogramot, valahányszor célszerűnek találod. Törekedj hatékony megoldásra az algoritmusok időigénye tekintetében. **Lásd el beszédes kommentekkel programjaidat**.
- A bemeneti adatok helyesnek tekinthetők.

FELADATOK:

1. Legyen az alábbi pszeudokód programrészlet:

```
eredmeny = 0
┌amíg x ≠ 0 végezd
| eredmeny = eredmeny * 10 + 9 - (x % 10)
| x = x / 10
└■
kiír eredmeny
```

Mit ír ki a fenti programrészlet, ha `x = 20230506`? [1 pont]

a) 79769493
b) 39496797
c) 60503202
d) 20230506

2. Legyen az alábbi pszeudokód programrészlet:

```
eredmeny = 0
┌amíg x ≠ 0 végezd
| ┌ha (x % 10) % y == 0 akkor
| |  eredmeny = eredmeny * 10 + x % 10
| └■
| x = x / 10
└■
kiír eredmeny
```

Mit ír ki a fenti programrészlet, ha `x = 20230506` és `y = 3`? [1 pont]

a) 3006
b) 60030
c) 225
d) 522

3. Legyen az alábbi pszeudokód programrészlet:

```
eredmeny = 0
p = 1
┌amíg x ≠ 0 végezd
|  ┌amíg x ≠ 0 és x % 2 == 0 végezd
|  |   eredmeny = eredmeny * 10 + x % 10
|  |   p = p * 10
|  |   x = x / 10
|  └■
|  ┌amíg x ≠ 0 és x % 2 == 1 végezd
|  |   eredmeny = eredmeny + ( x % 10 ) * p
|  |   p = p * 10
|  |   x = x / 10
|  └■
└■
kiír eredmeny
```

Mit ír ki a fenti programrészlet, ha `x = 20230506`? [1 pont]

a) 600202
b) 600202356
c) 356
d) 35600202
e) 20200635
f) 53202006

4. Legyen az alábbi pszeudokód programrészlet, ahol a `prim(n)` függvény igazat ad vissza, ha az `n` prímszám, ellenkező esetben hamisat:

```
szamol = 0
┌minden i = 1, n végezd
|   x = a[i]
|   eredmeny = 0
|  ┌amíg x ≠ 0 akkor
|  |  ┌ha prim(x % 10) akkor
|  |  |   eredmeny = eredmeny * 10 + x % 10
|  |  └■
|  |   x = x / 10
|  └■
|  ┌ha eredmeny == 0 akkor
|  |    szamol = szamol + 1
|  |különben
|  |    kiír a[i], ' ', eredmeny, '\n'
|  └■
└■
kiír szamol
```

Mit ír ki a fenti programrészlet, ha az `a` tömb tartalma: `2345, 7902, 86420, 88, 2023, 122, 4` ebben a sorrendben, és `n = 7`? [1 pont]

7902 27
86420 2
2023 322
122 22
2

5. Írj Pascal vagy C/C++ programot, amely bekéri a felhasználó korát (`1≤kor≤120`), majd megvizsgálja, hogy a felhasználó fiatalkorú-e (18 év alatti), felnőtt-e (18-65 év közötti) vagy nyugdíjas-e (65 év feletti). [1 pont]

Példa bement:

23
Kimenet:
felnőtt

```cpp
#include <iostream>

using namespace std;

int main() {
    int age;
    cin >> age;

    if (age < 18) {
        cout << "fiatalkoru";
    } else if (age > 65) {
        cout << "nyugdijas";
    } else {
        cout << "felnott";
    }

    return 0;
}
```

6. Írj Pascal vagy C/C++ programot, amely beolvas egy `n` (`2<n<100`) értéket, valamint egy `n` elemű egész számokat tartalmazó számsorozatot, majd kiírja a képernyőre azt az indexet, ahol a számsorozatban először fordul elő olyan szám, amely nagyobb, mint az előtte és az utána következő számok számtani közepe (átlaga). Ha ilyen szám nem található a számsorozatban, a program ezt írja ki a képernyőre „`Nincs ilyen szám`” üzenet formájában. Csak azokat a számokat teszteljük, amelyeknek létezik mindkét szomszédja, tehát az első és utolsó értéket nem kell vizsgálni. Az indexelést végezzük 1-től. [1 pont]

Példa bemenet:
7
4 3 2 5 7 6 8
Kimenet:
4

Magyarázat:
A legelső szám, amely szomszédjainak számtani közepe kisebb, mint a szám, az az 5-ös szám. Ez pedig a számsorozat 4. eleme (az index 1-től kezdődik).

4 → nincs kisebb szomszédja, nem vizsgáljuk
3 → (4+2)/2 = 3, 3 pedig nem kisebb, mint maga a szám (3), tehát ez nem jó megoldás
2 → (3+5)/2 = 4, nem jó, mert 4>2
5 → (2+7)/2 = 4.5 , megfelelő, mert 4.5>5, tehát ennek a számnak az indexe 4, azaz a negyedik a számok között, tehát ez kerül kiírásra a képernyőn. Mivel megtaláltuk az első ilyen számot, tovább nem szükséges folytatni a keresést.

```cpp
#include <iostream>

using namespace std;

int main() {
    // beolvasás
    int n;
    cin >> n;

    // bal oldalsó, középső, jobb oldali szám
```

```
    int left, mid, right;

    for (int i = 0; i < n; i++) {
        // átcseréljük a számokat a memóriában
        left = mid;
        mid = right;

        // beolvassuk a következő számot a jobb oldalra
        cin >> right;

        // ha beolvastunk elég számot,
        // utána ellenőrizzük az átlagot
        if (i > 1 && mid > (left + right) / 2.0) {
            // ha volt a feltételnek megfelelő szám
            cout << "Az elso ilyen elem indexe: " << i << endl;
            return 0;
        }
    }

    // ha nem volt ilyen érték
    cout << "Nincs ilyen szam." << endl;
    return 0;
}
```

7. Egy `nxn`-es (`2<n<100`), négyzet alakú parkot mindenhol fűvel ültettek be. Ezután felparcellázták egyforma négyzetekre és néhányukba egy vagy több gyümölcsfát, illetve díszfát ültettek. Ismert, hogy 4 fajta gyümölcsfából és ugyancsak 4 fajta díszfából válogathattak. Minden parcellát egy pontosan 8 számjegyű természetes számmal kódoltak, ha ültettek legalább 1 fát a parcellára és 0-val, ha egyetlen fát sem ültettek arra a parcellára. A 8 számjegyű szám első 4 számjegye külön-külön azt jelképezte, hogy hány gyümölcsfát ültettek egy-egy fajtából, az utolsó 4 számjegye pedig, hogy hányat ültettek egy-egy fajta díszfából.
Példa:

- a 20355644 kód jelentése: arra a parcellára 3 különböző fajta gyümölcsfát ültettek, összesen 10-et, illetve 4 fajta díszfát ültettek, ezekből összesen 19-et.

Írj egy C/C++ programot, amely beolvassa a park.txt állományból a park tervrajzát (első sorban egy `n` érték található, utána `nxn` érték) és kiírja a képernyőre [1 pont]:

a) Hány gyümölcsfát, illetve hány díszfát ültettek összesen a parkba
b) Hány olyan parcella van, amelyen nem csak fű van, ugyanannyi fajta gyümölcsfát, mint díszfát tartalmaz és mind a 8 szomszédos parcelláján csak fű található?

Bemenet (park.txt):
4

12100000	0	0	0
34511011	0	**10012020**	0
33331201	0	0	0
54000000	0	79118679	0

Eredmény:
a) gyümölcsfák száma: $4 + 13 + 2 + 12 + 9 + 18 = \mathbf{58}$
díszfák száma: $3 + 4 + 4 + 30 = \mathbf{41}$
b) 1 ilyen parcella van (az **10012020** kód azt mutatja, hogy ezen a parcellán 2 fajta gyümölcsfát és ugyancsak 2 fajta díszfát ültettek és mind a 8 szomszédján fű van)

```
#include <iostream>
#include <fstream>
```

```cpp
using namespace std;

int main() {
    // Az adatok beolvasása a park.txt állományból
    int n;
    ifstream fin("park.txt");
    fin >> n;
    int park[n][n];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            fin >> park[i][j];
        }
    }

    fin.close();

    // Összesítjük a gyümölcsfák és díszfák számát
    int gyumolcsfakszama = 0, diszfakszama = 0, gyumolcsfakfajta = 0, diszfakfajta = 0,
eredmeny = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int gyumolcsfak[4], diszfak[4];
            gyumolcsfakfajta = diszfakfajta = 0;
            int kod = park[i][j];

            for (int k = 0; k < 4; k++) {
                diszfak[k] = kod % 10;
                kod /= 10;
            }

            for (int k = 0; k < 4; k++) {
                gyumolcsfak[k] = kod % 10;
                kod /= 10;
            }

            for (int k = 0; k < 4; k++) {
                diszfakszama += diszfak[k];
                gyumolcsfakszama += gyumolcsfak[k];

                if (gyumolcsfak[k] != 0)
                    gyumolcsfakfajta++;
                if (diszfak[k] != 0)
                    diszfakfajta++;
            }

            // Ellenőrizzük a szomszédos parcellákon, hogy csak fű legyen
            if (i >= 1 && j >= 1) {
                bool van = true;

                if ((gyumolcsfakfajta == diszfakfajta && (gyumolcsfakfajta != 0) &&
(park[i][j] != 0))) {
                    for (int k = -1; k <= 1; k++) {
                        for (int l = -1; l <= 1; l++) {
```

```
                            if (k == 0 && l == 0) continue;  // A saját parcellánkat
kihagyjuk
                            if (park[i + k][j + l] != 0) {
                                van = false;
                                break;
                            }
                        }

                        if (!van) break;
                    }

                    if (van) {
                        eredmeny++;
                        cout << i << " " << j << " " << endl;
                    }
                }
            }
        }
    }

    cout << "gyumolcsfak szama: " << gyumolcsfakszama << endl;
    cout << "diszfak szama: " << diszfakszama << endl;
    cout << "Eredmeny: " << eredmeny << " parcella" << endl;

    return 0;
}
```

8. Egy szöcske-pár azt vette fejébe, hogy átszökdelnek egy négyzetes mátrixon: az egyik sorról sorra, a másik pedig oszlopról oszlopra. Közös elhatározásuk, hogy maximumról maximumra fognak szökdelni. Írj programot, amely beolvassa a matrix.txt állományból az n értéket (`0<n<100`), majd az `nxn` méretű mátrixot (minden eleme pozitív egész), és kiírja a képernyőre, hogy a szöcske-pár hány „forgatókönyv” közül választhat (lásd a példát). [1 pont]

Példa bemenet (matrix.txt):
4
9 6 8 3
9 17 2 1
9 3 21 8
9 2 4 33

Kimenet:
16

Magyarázat:
1 útvonal sorról sorra: (1,1)(2,2)(3,3)(4,4)
4 útvonal oszlopról oszlopra: (1,1)(2,2)(3,3)(4,4); (2,1)(2,2)(3,3)(4,4); (3,1)(2,2)(3,3)(4,4); (4,1)(2,2)(3,3)(4,4)
Mivel minden útvonal két irányból szökdelhető végig, ezért a forgatókönyvek száma $2 * 8 = 16$.

```
#include <iostream>
#include <fstream>
#define N 100

using namespace std;

int forgatokonyvek_szama(int a[][N], int n);
```

```cpp
int main() {
    int n, a[N][N];
    ifstream fin("matrix.txt");
    fin >> n;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            fin >> a[i][j];
        }
    }

    cout << forgatokonyvek_szama(a, n);
    return 0;
}

int forgatokonyvek_szama(int a[][N], int n)
{
    int maxKurrensSor, maxKurrensOszlop, maxKurrensSorDb,
maxKurrensOszlopDb;
    int forgatokonyvek = 1;

    for (int i = 0; i < n; ++i) {
        maxKurrensSor = maxKurrensOszlop = 0;

        for (int j = 0; j < n; ++j) {
            if (a[i][j] > maxKurrensSor) {
                maxKurrensSor = a[i][j];
                maxKurrensSorDb = 1;
            } else if (a[i][j] == maxKurrensSor) {
                ++maxKurrensSorDb;
            }

            if (a[j][i] > maxKurrensOszlop) {
                maxKurrensOszlop = a[i][j];
                maxKurrensOszlopDb = 1;
            } else if (a[j][i] == maxKurrensOszlop) {
                ++maxKurrensOszlopDb;
            }
        }

        forgatokonyvek *= maxKurrensSorDb;
        forgatokonyvek *= maxKurrensOszlopDb;
    }

    return 4 * forgatokonyvek;
}
```

9. Egy családfának csak az apai ágát úgy ábrázoljuk, hogy mindegyik benne levő személynek van egy egyedi azonosítója (`1, 2, …, n` számsorozat) és tároljuk a születési évét. Írj egy C/C++ programot, amely beolvassa a csaladfa.txt állományból a családfát, majd megadja [1 pont]:

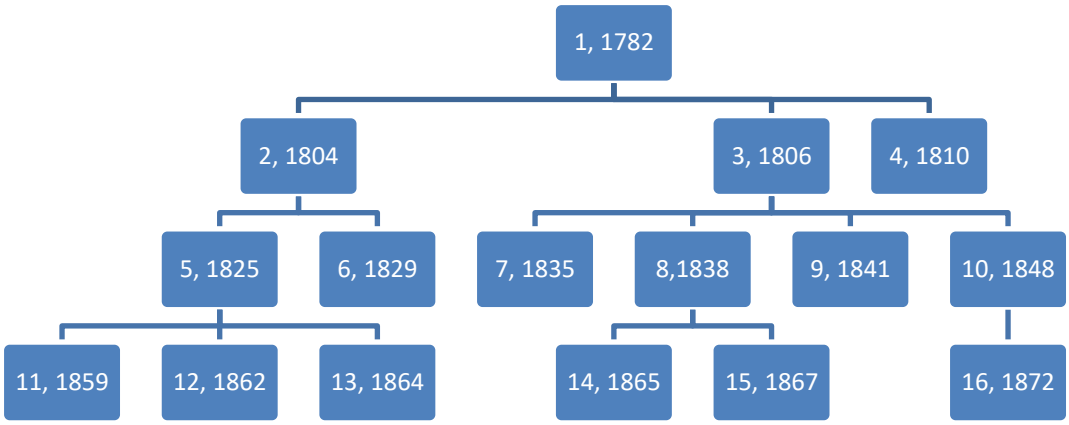
a) A gyökér (legkorábbi ős) és a levelekben lévő személyek (leszármazottak) közötti legnagyobb és legkisebb korkülönbséget.
b) A levelekben lévő személyek életkorának átlagát a mai dátumon.

A bemenet első sora a személyek számát tartalmazza (`0<n<100`). Ezt követően az állomány minden sora

három egész értéket tartalmaz, az első a személy egyedi azonosítója, a második a személy apjának az azonosítója (ez 0, ha a gyökérben vagyunk), a harmadik pedig a születés éve.

Példa bemenet (csaladfa.txt):

16		
1	0	1782
2	1	1804
3	1	1806
4	1	1810
5	2	1825
6	2	1829
7	3	1835
8	3	1838
9	3	1841
10	3	1848
11	5	1859
12	5	1862
13	5	1864
14	8	1865
15	8	1867
16	10	1872



Eredmény:

c) Legnagyobb korkülönbség: 90
   Legkisebb korkülönbség: 28
d) Átlagéletkor: 172,6

```
#include <iostream>
#include <fstream>
#define N 100

using namespace std;

typedef struct {
     int ev;
     bool apa;
} SZEMELY;

SZEMELY a[N];

int main() {
     int n, id, apa, szuletesiev, gyoker;
     ifstream fin("csaladfa.txt");
     fin >> n;
```

```
    for (int i = 0; i < n; ++i) {
        fin >> id >> apa >> szuletesiev;

        if (apa == 0) gyoker = id;

        a[apa].apa = true;
        a[id].ev = szuletesiev;
    }

    int maxKorkulonbseg = 0, minKorkulonbseg = 2023, levelDb = 0;
    float LevelKorOsszeg = 0;

    for (int i = 1; i <= n; ++i) {
        if (a[i].apa) continue;

        // i. levél ID
        ++levelDb;
        LevelKorOsszeg += (2023 - a[i].ev);

        if (a[i].ev - a[gyoker].ev > maxKorkulonbseg) {
            maxKorkulonbseg = a[i].ev - a[gyoker].ev;
        }

        if (a[i].ev - a[gyoker].ev < minKorkulonbseg) {
            minKorkulonbseg = a[i].ev - a[gyoker].ev;
        }
    }

    cout << "Legnagyobb korkulonbseg: " << maxKorkulonbseg << endl;
    cout << "Legkisebb korkulonbseg: " << minKorkulonbseg << endl;
    cout << "Atlageletkor: " << LevelKorOsszeg / levelDb << endl;

    return 0;
}
```