INFORMATIKA – tétel
2018

ELIGAZÍTÁS:

- 1 pont hivatalból;
- Az 1-4 feladatokban (a pszeudokód programrészletekben): (1) a **`kiír '\n'`** utasítás újsorba ugratja a képernyőn a kurzort; (2) a **`minden i = n,1,-1 végezd`** programsor jelentése, hogy minden `i = n,n-1,n-2,…,1` értékre…
- 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:

```
x[1] = 1
┌minden i = 2,n végezd
| x[i] = x[i-1] + 2 * i - 1
| kiír x[i],',',i*i,'\n'
└■
```

Mit ír ki a fenti programrészlet, ha `n=7`? [1 pont]

```
4,4
9,9
16,16
25,25
36,36
49,49
```

2. Legyen az alábbi pszeudokód programrészlet:

```
┌minden i = 1,n végezd
| ┌minden j = 1,i végezd
| | ┌minden k = 1,i végezd
| | | x[i][j][k] = i
| | └■
| └■
└■
kiír x[n][n][n]
```

Mit ír ki a fenti programrészlet, ha `n=4`? [1 pont]

```
4
```

3. Legyen az alábbi pszeudokód programrészlet:

```
┌minden i = 1,n végezd
| ┌minden j = 1,i végezd
| | ┌minden k = 1,i végezd
| | | ┌ha j>k akkor
| | | |  kiír j
| | | | különben
| | | |  kiír k
| | | └■
| | └■
```

```
| | kiír '\n'
| ┗■
┗■
```

Mit ír ki a fenti programrészlet, ha n=4? [1 pont]

```
1
12
22
123
223
333
1234
2234
3334
4444
```

4. Legyen az alábbi pszeudokód programrészlet:

```
x[0] = 1
y = 1
┌minden i = 1,n végezd
| y = y * 11
| x[i] = 1
| ┌minden j = i-1,1,-1 végezd
| | x[j] = x[j] + x[j-1]
| ┗■
| kiír y,','
| ┌minden j = 0,i végezd
| | kiír x[j]
| ┗■
| kiír '\n'
┗■
```

Mit ír ki a fenti programrészlet, ha n=4? [1 pont]

```
11,11
121,121
1331,1331
14641,14641
```

5. Legyen egy n érték (0≤n≤10), valamint egy n+1 elemű valós számsorozat, amelynek elemei egy n-ed fokú polinom együtthatóit jelentik. Írj Pascal vagy C/C++ programot, amely billentyűzetről beolvassa az n értéket, az n+1 elemű számsorozatot, majd kiírja a képernyőre, hogy hány tagú a polinom (hány együtthatója nem nulla). [1 pont]

```
#include <iostream>

int main()
{
    int n;

    std::cout << "Enter n: ";
    std::cin >> n;

    std::cout << "Enter coefficients: ";

    int count = 0;
    int coeff = 0;

    for(int i = 0; i < n+1; i++)
    {
```

```
        std::cin >> coeff;

        if (coeff != 0)
        {
            count++;
        }
    }

    std::cout << "Non-zero coefficient count: " << count << std::endl;
}
```

6. Legyen egy n érték ($0 \le n \le 10$), valamint egy n+1 elemű valós számsorozat, amelynek elemei egy n-ed fokú polinom együtthatóit jelentik. Az együtthatók a tagok fokszámának növekvő sorrendjében vannak megadva. Írj Pascal vagy C/C++ függvényt, amely paraméterként megkapja az n értéket, az n+1 elemű számsorozatot, illetve egy további x valós értéket. A függvény térítse vissza a polinom helyettesítési értékét az x helyen. [1 pont]

```
#include <stdio.h>
#include <stdlib.h>

float szamol(int n, float t[], float x){
      float p = 1,e = 0;
      int i;
      for ( i = 0 ; i <= n ; i++ ){
        if ( t[i] != 0 ){
            e += t[i]*p;
        }
        p *= x;
      }
      return e;
}

int main(){
      int n,i;
      float t[11],x;
      printf("n="); scanf("%i", &n);
      printf("x="); scanf("%f", &x);
      for( i = 0 ; i <= n ; i++ ){
        printf("f[%i]=", i);
        scanf("%f", &t[i]);
      }
      printf("%0.2f", szamol(n,t,x));
      return 0;
}
```

7. Írj Pascal vagy C/C++ programot, amely a matrix.txt állományból beolvassa az n értéket ($3 \le n \le 51$), valamint egy nxn méretű mátrixot, majd kiír egy üzenetet aszerint, hogy a mátrix bűvös négyzet-e. (Egy nxn méretű bűvös négyzetet úgy definiálunk, mint amelynek (1) elemei a $1, 2, 3, \ldots, n^2$ számsorozat elemei, (2) elemei páronként különböznek, (3) minden sora, oszlopa, valamint a fő- és mellék-átlója mentén ugyanannyi az összeg) [1 pont]

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#define N 51

using namespace std;

int n;
int matrix [ N ][ N ];
// elofordulasok ellenorzesere
```

```
bool exist[ N*N + 1]{false};

void readMatrix(const char * filename);
bool isMagicSquare();

int main() {
    readMatrix("matrix.txt");
    if (isMagicSquare()) {
        cout << "Magic square" << endl;
    } else {
        cout << "Not magic square" << endl;
    }
    return 0;
}

void readMatrix(const char * filename) {
    ifstream ifs(filename);
    if (!ifs) {
        cout << "File megnyitasi hiba" << endl;
        exit(1);
    }
    ifs >> n;
    if (n < 3 || n > 51) {
        cout << "n erteke nem megfelelo " << n << endl;
        exit(1);
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            ifs >> matrix[ i ][ j ];
        }
    }
}

bool isMagicSquare () {
    long sumRow[ n ]{0};
    long sumCol[ n ]{0};
    long sumMainDiag = 0;
    long sumSecondDiag = 0;
    long total = (n * (n * n + 1)) / 2;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            // ha nem megfelelo szam
            if (matrix[ i ][ j ] < 1 || matrix[ i ][ j ] > n * n) {
                return false;
            }
            // ha mar elofordult
            if (exist[ matrix[ i ][ j ] ] == true) {
                return false;
            }
            exist[ matrix[ i ][ j ] ] = true;
            sumRow[ i ] += matrix[ i ][ j ];
            sumCol[ j ] += matrix[ i ][ j ];
            if (i == j) {
                sumMainDiag += matrix[ i ][ j ];
            }
            if ( i + j == n - 1) {
                sumSecondDiag += matrix[ i ][ j ];
            }
        }
    }
    for( int i=0; i<n; ++i){
        if( sumRow[ i ] != total ){
            return false;
```

```
        }
        if( sumCol[ i ] != total ){
            return false;
        }
    }
    if( sumMainDiag != total || sumSecondDiag != total  ){
        return false;
    }
    return true;
}
```

8. Ahogy az előző feladatból is megtudhattuk, egy bűvös négyzet egy olyan négyzetes mátrix, amelynek minden sora, oszlopa és mindkét átlója mentén ugyanannyi az összeg. Írj Pascal vagy C/C++ programot, amely beolvas a billentyűzetről egy p értéket ($1 \le p \le 25$), majd generál (és kiírja állományba) egy $(2p+1) \times (2p+1)$ méretű bűvös négyzetet a Loubère-algoritmus szerint. E módszer a következőképpen tölti fel a mátrixot az $1, 2, 3, \ldots, (2p+1)^2$ számsorozattal:

- az első sor közepére 1-est teszünk, majd átlósan jobbra-fel irányba haladunk;
- a mátrixot körkörösnek tekintjük, azaz úgy, mint amelyben a legfelső sort, felfele irányba, a legalsó követ, és a legjobboldaliabb oszlopot, jobbra irányba, a legbaloldaliabb követ;
- ha az a pozíció, amely átlósan jobbra-fel irányba következne, már foglalt, akkor a kurrens pozíció alatti cellával folytatjuk a kitöltést. (Lásd a példát $p=2$-re) [1 pont]

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int p;
    printf("p = ");
    scanf("%d",&p); ///beolvasas
    int n=2*p+1, limit = n*n; /// inicializalas
    int x=0, y=n/2, a, b; /// kezdopozicio es ellenorzopontok

    ///helyfoglalas - statikusan vagy dinamikusan - nullaval feltoltott tomb
    ///int a[51][51]={0}   ///vagy
    int **t = (int*)calloc(n,sizeof(int*));
    for (int i = 0; i < n; ++i)
        t[i] = (int*)calloc(n,sizeof(int));

    /// A lepesszam ismert, a korkorosseget a modulo muvelet biztositja
    for (int counter = 1; counter <= limit; ++counter){
        t[x][y] = counter;
        a = (x - 1 + n) % n; ///hozzaadunk n-et, hogy nehogy negativ szam legyen
        b = (y + 1) % n;
        if (t[a][b]){  /// ha mar foglalt a jobbra fel hely, akkor lefele lep
            x = (x + 1) % n;
        }
        else  ///egyebkent jobbra fel
        {
            x = a;
            y = b;
        }
    }

     ///kiirjuk az eredmenyeket
   if(!freopen("ki.txt","wt",stdout)){
```

```
        printf("Bemeneti hiba!");
        return 0;
    }

    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j)
            printf("%4d", t[i][j]);
        printf("\n");
    }

    for (int i = 0; i < n; ++i){
        free(t[i]);
    }
    free(t);

    freopen("CON","wt",stdout);

    return 0;
}
```

9. N személy evezős csónakkal (nem működik távirányítóval) szeretne átkelni egy folyón. Írj Pascal vagy C/C++ programot, amely állományból beolvassa (az adatok szóközzel vannak elválasztva), hogy mekkora tömeget bír el a csónak ($1 \le G \le 200$), a személyek számát ($1 \le n \le 2000000000$), és ezek tömegértékeit ($1 \le g_i \le 200,\ i=1,n$), majd kiír a képernyőre egy `IGEN` vagy `NEM` üzenetet aszerint, hogy van, vagy nincs lehetőség az átkelésre. [1 pont]

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    if(!freopen("be.txt","rt",stdin)){
        printf("Bemeneti hiba!");
        return 0;
    }
    int csonak_max, n, min1, min2, seged, x;
    min1 = min2 = 201;  /// inicializalunk a (max(tomeg) + 1)-el
    scanf("%d%d", &csonak_max, &n);

    /// Akkor tudnak atkelni a folyon, ha
    ///      csak egy ember van, es az befer a csonakba,
    ///      vagy
    ///      ( van legalabb ket ember, aki egyutt tud csonakazni / min1, min2
    ///        es
    ///        a legsulyosabb ember is at tud kelni a csonakkal)

    for (int i=0;i<n;++i){
        scanf("%d", &x);
        if (x > csonak_max){
            printf("NEM"); /// Ha tul sulyos az ember, nem tud atmenni
            return 0;      /// a csonakkal, akkor nincs megoldas
        }
        /// manualis beszuro rendezes, a ket legkonnyebb embert nyilvantartom
        if (x < min2) {
            min2 = x;
        }
        if (min2 < min1) {
            seged = min1;
            min1 = min2;
            min2 = seged;
        }
    }
    fclose(stdin);
```

```
    if (n == 1 || min1 + min2 <= csonak_max)
        printf("IGEN");  /// csak egy ember van,
                         /// vagy a ket legkonnyebb ember at tud menni egyutt
    else
        printf("NEM");   /// egyebkent nincs megoldas
    return 0;
}
```