

# emter

## INFORMATIKA – tétel

### 2024

#### ELIGAZÍTÁS:

- 1 pont hivatalból;
- Az 1-4 feladatokban (a pszeudokód programrészletekben): (1) a kiírások szóközzel elválasztva történnek; (2) a „/” operátor osztási hányadost ad meg, a „%” operátor pedig osztási maradékot.
- Az 5-9 feladatok esetében használj alprogramot, valahányszor célszerűnek találsz. Maximális pontszám **hatékony megoldásra** kapható (elsődleges szempont az algoritmusok időigénye). Lásd el **beszédes kommentekkel** programjaidat.
- A bemeneti adatok helyesnek tekinthetők. Minden egész szám eltárolható az `int` tartományban.

#### FELADATOK:

1. Legyen az alábbi pszeudokód programrészlet:

```
eredmeny = 0
┌amíg x%10 ≠ 0 végezd
| eredmény = eredmény + 1
| x = x / 10
└─
┌amíg x/10 ≠ 0 végezd
| eredmény = eredmény - 1
| x = x / 10
└─
kiír eredmény
```

Mit ír ki a fenti programrészlet, ha  $x = 27240576$ ? [1 pont]

-1

2. Legyen az alábbi pszeudokód programrészlet:

```
L0 = 2
L1 = 1
F0 = 0
F1 = 1
┌amíg L1 - F1 < 100 végezd
| kiír L1 - F1
| x = L1
| y = F1
| L1 = L0 + L1
| F1 = F0 + F1
| L0 = x
| F0 = y
└─
```

Mit ír ki a fenti programrészlet? [1 pont]

0 2 2 4 6 10 16 26 42 68

3. Legyen az alábbi pszeudokód programrészlet:

```
x = 1
y = n
┌amíg x ≤ y végezd
| kiír a[(x+y)/2]
| ┌ha a[(x+y)/2] ≥ 0 akkor
| | x = (x+y)/2+1
```

```

| |különben
| | y = (x+y)/2-1
| |
| |
| |

```

Mit ír ki a fenti programrészlet, ha  $n=13$  és  $a[1..n]=\{-13, 0, 4, -6, 8, 5, 1, 8, -8, -3, 2, 1, -4\}$ ? [1 pont]

- a) 1 -3 8
- b) 1 -3 8 -8**
- c) 1 -3 -8
- d) 1 2 1

4. Legyen az alábbi pszeudokód programrészlet:

```

minden i = 0, n*m-1 végezd
| a[i/m][i%m] = i
|
kiír a[0][0], a[0][m-1], a[n-1][0], a[n-1][m-1]

```

Mit ír ki a fenti programrészlet, ha az  $n=5$  és  $m=7$ ? [1 pont]

**0 6 28 34**

5. Írj Pascal vagy C/C++ *programot*, amely billentyűzetről bekéri egy termék árát (természetes szám) és kiírja a képernyőre az alkalmazandó kedvezményt az alábbi táblázat alapján: 0-1000 lej: 0% kedvezmény, 1001-5000 lej: 5% kedvezmény, 5001-10000 lej: 10% kedvezmény, 10000 lej felett: 15% kedvezmény.

[1 pont]

Példa bemenet:

10000

Kimenet:

10% kedvezmény

```

int ar;
cin >> ar;
if( ar < 1001 ){ cout << "0% kedvezmeny"; }
else if( ar < 5001 ){ cout << "5% kedvezmeny"; }
else if( ar < 10001 ){ cout << "10% kedvezmeny"; }
else{ cout << "15% kedvezmeny"; }

```

6. Írj Pascal vagy C/C++ *függvényt*, amely paraméterként megkap egy tömböt (legyen x), és a tömb tárolta egész számsorozat hosszát (legyen n), és visszatérít 1-et vagy 0-t aszerint, hogy a számsorozat elemei azonosak vagy sem. [1 pont]

Lehetséges C/C++ függvényfejléc:

```
int fuggv( int x[], int n )
```

Példa bemenet:

n = 7

x = [4, 3, 4, 4, 4, 4, 4]

Kimenet:

0

```

int fuggv( int x[], int n ){
    for( int i = 1 ; i < n ; ++i ){
        if( x[i] != x[0] ){ return 0; }
    }
    return 1;
}

```

7. Írj Pascal vagy C/C++ *programot*, amely  $[0, 10)$  intervallumba eső valós számokat olvas be a *szamok.txt* állományból (legfentebb 1000 számot), majd a számokat az egészrészük alapján egymásalatti sorokba csoportosítva írja ki a *csoportok.txt* állományba. A sorok egészrész szerint növekvő sorrendbe legyenek

rendezve, és egy soron belül a számok sorrendje egyezzen meg a bementi állománybéli sorrendjükkel (lásd a példát) [1 pont]

Példa bement (*szamok.txt*):

```
1.77 5.6 0.4 6.67 3.75 5.34 2.7 2.9
1.9 2.4 0.33 8.44
6.78 3.79 8.56
9.75 2.45 5.43
```

Kimenet (*csoportok.txt*):

```
0.4 0.33
1.77 1.9
2.7 2.9 2.4 2.45
3.75 3.79
5.6 5.34 5.43
6.67 6.78
8.44 8.56
9.75
```

```
int main(){
    double x, arr[10][1000];
    int n[10] = {0};
    ifstream fin("szamok.txt");
    ofstream fout("csoportok.txt");
    while( fin >> x ){
        int i = x;
        arr[i][n[i]] = x; ++n[i];
    }
    for( int i = 0 ; i < 10 ; ++i ){
        if( n[i] == 0 ){ continue; }
        for( int j = 0 ; j < n[i] ; ++j ){
            fout << arr[i][j] << " ";
        }
        fout << endl;
    }
    return 0;
}
```

8. Írj Pascal vagy C/C++ *függvényt*, amely paraméterként megkap egy tömböt (legyen  $x$ ), és a tömb tárolta valós számsorozat hosszát (legyen  $n$ ), és visszatérít annak az *összefüggő résztömbnek* az elemösszegét, amely esetében ez az összeg maximális. Összefüggő résztömb:  $x_j, x_{j+1}, \dots, x_{j+k-1}$ . [1 pont]

Lehetséges C/C++ függvényfejléc:

```
double max_reszosszeg( double x[], int n )
```

Példa bemenet:

```
n = 10
x = [31, -41, 59, 26, -53, 58, 97, -93, -23, 84]
```

Kimenet:

```
187
```

```
double max_reszosszeg(double x[], int n) {
    double maxx = 0;
    double max_pozicioig = 0;
    for (int i = 0; i < n; i++) {
        max_pozicioig = max(max_pozicioig + x[i], 0.0);
        maxx = max(maxx, max_pozicioig);
    }
    return maxx;
}
```

9. Írj Pascal vagy C/C++ *függvényt*, amely paraméterként megkap egy tömböt (legyen  $x$ ), és a tömb tárolta bináris számsorozat hosszát (legyen  $n$ ). Tekintsük úgy, hogy a bináris számsorozat izzók sorozatát ábrázolja, ahol a 0 kikapcsolt, az 1 pedig bekapcsolt izzót jelöl. Az izzókat úgy szeretnénk átrendezni, hogy az összes 0-ás érték a tömb bal oldalán, az összes 1-es érték pedig a tömb jobb oldalán legyen csoportosítva. A tömb bármely két eleme között cserét végezhethetünk, és megelőlegezhetjük, hogy a sorozat mindenképp tartalmaz

legalább egy 0-ás és 1-es értéket. A függvény térítse vissza a minimális csereszámot, amely szükséges ahhoz, hogy az izzók a fent leírt sorrendbe kerüljenek (0-k bal oldalt, 1-ek jobb oldalt). [1 pont]

Lehetséges C/C++ függvényfejléc:

```
int min_csereszam( int x[], int n )
```

Példa bemenet:

```
n = 12
```

```
x = [1,0,1,0,1,1,0,0,1,0,1,1]
```

Kimenet:

```
3
```

```
int min_csereszam (int nums[], int n){
    int minSwaps = 0;           // csereszám
    int left = 0;              // pointer balra
    int right = n - 1;         // pointer jobbra
    // amíg a két pointerünk össze nem ér
    while (left < right) {
        // megkeressük a legjobboldalibb 1-est
        while (nums[left] == 0) {
            left++;
        }
        // megkeressük a legbaloldalibb 0-ást
        while (nums[right] == 1) {
            right--;
        }
        // ha a két pointer "nem ért össze",
        // azaz a bal pointer még mindig bal oldalt van,
        // akkor felcseréljük az elemeket
        if (left < right) {
            swap(nums[left], nums[right]);
            minSwaps++;
        }
    }
    return minSwaps;
}
```