Sapientia Erdélyi Magyar Tudományegyetem
Marosvásárhelyi Kar

Felvételi, 2025. szeptember
– Alapképzés, informatika vizsga –

Hivatalból 1 pont jár. Munkaidő 3 óra.
Megjegyzések:

- A pszeudokód algoritmusban
  - Az „=” értékadást, az „==” pedig egyenlőség vizsgálatot jelent
  - A / operátorral osztási hányadost, a % operátorral pedig osztási maradékot kapunk meg
  - A „`minden i = 1, n végezd`” jelentése: minden $i = 1, 2, 3, \ldots, n$ értékre végezd el…
  - A „`minden i = n, 1, -1 végezd`” jelentése: minden $i = n, n\text{-}1, n\text{-}2, \ldots, 1$ értékre végezd el…
- **A programokat megjegyzésekkel kell ellátni!**
- **Törekedjünk hatékony megoldásokra!** (maximális pontszám feltétele)

1. (1 pont) Legyen az alábbi pszeudokód programrészlet:

```
res = 0
┌amíg n ≠ 0 végezd
| res = res + (n % 3)
| n = n / 10
└■
kiír res
```

Mit ír ki a fenti programrészlet, ha $n = 20240729$, illetve ha $n = 1315179$?

```
13
7
```

2. (1 pont) Legyen az alábbi pszeudokód programrészlet:

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

Mit ír ki a fenti programrészlet, ha $x = 368238$, $y = 3$ és $z = 8$?

```
8383
```

3. (1 pont) Legyen az alábbi pszeudokód programrészlet, ahol a prim(x) függvény igazat ad vissza, ha x prím szám, ellenkező esetben hamisat:

```
count = 0
res = ...
┌minden i = 1, n végezd
| ┌ha prim(a[i]) akkor
| |   count = count + a[i]%2
| |  ┌ ha a[i] < res akkor
| |  |  res = a[i]
| |  └■
| └■
|
└■
kiír count, ' ', res
```

a) Mi kellene kerüljön a pontok helyére, ha azt szeretnénk, hogy a *res* változó az algoritmus végrehajtása után a legkisebb, $a[1..n]$ tömbben szereplő prím értéket tárolja. FONTOS: tudjuk, hogy az utolsó elem garantáltan prím.

b) Mit ír ki a fenti programrészlet (az *a)* pont szerinti értékadás mellette), ha az $a[1..n]$ tömb

tartalma:

i. 15, 3, 2, 88, 16, 34, 11 ebben a sorrendben, és $n = 7$?
ii. 18, 32, 22, 11, 7 ebben a sorrendben, és $n = 5$?

```
a[n]
2 2
2 7
```

4. (1 pont) Legyen az alábbi pszeudokód programrészlet:

```
minden i = n, 2, -1 végezd
    minden j = n, 2, -1 végezd
        ha m[i+1][j] < m[i][j+1] akkor
            m[i][j] = m[i+1][j]
        különben
            m[i][j] = m[i][j+1]
```

Mi lesz az $m[1..n][1..n]$ kétdimenziós tömb tartalma, ha kezdetben az utolsó sora az 1, 3, 7, 2, 4 és utolsó oszlopa pedig az 1, 8, 3, 2, 4 értékeket tartalmazza, ebben a sorrendben, és $n = 5$?

```
1 1 1 1 1
1 2 2 2 8
1 2 2 2 3
1 2 2 2 2
1 3 7 2 4
```

Helyesnek fogadtuk el azt is, ha valaki hibás kódot jelzett, ugyanis a minden ciklusok „minden i = n-1, 1, -1 végezd” alakúak kellet volna legyenek.

5. (1 pont) Írj Pascal vagy C/C++ **programot**, amely beolvas a billentyűzetről egy természetes számot, amely vagy 65 és 90 közé, vagy 97 és 122 közé esik? Ha a szám 65 és 90 között van, akkor adj hozzá 32-t, ha pedig 97 és 122 között van, akkor vonjál ki belőle 32-t. A program írja ki a módosított értéket.

Példa:

Bemenet:

```
88
```

Kimenet:

```
120
```

```
int main(){
    int x;
    cin >> x;
    if ( x >= 65 && x <= 90 ){
        x += 32;
    }
    else if ( x >= 97 && x <= 122 ){
        x -= 32;
    }
    cout << x;
    return 0;
}
```

6. (1 pont) Írj Pascal vagy C/C++ **függvényt**, amely paraméterként megkapja az n értéket, valamint egy egydimenziós tömböt, amely egy n elemű számsorozatot tárol (az elemei természetes számok a [0,255] intervallumból). A függvény módosítsa a számsorozatot oly módon, hogy:

- minden elem értékét, amely a [65,90] intervallumba esik, növelje 32-vel;
- minden elem értékét, amely a [97,122] intervallumba esik, csökkentse 32-vel;
- minden más elem értéke maradjon változatlan.

Példa:

Bemenet:

```
5
12 65 98 200 66
```

Kimenet:

```
12 97 66 200 98
```

```
void fuggveny ( int n, int x [100] ){
    for ( int i = 0 ; i < n ; ++i ){
        if ( x[i] >= 65 && x[i] <= 90 ){
            x[i] += 32;
        }
        else if ( x[i] >= 97 && x[i] <= 122 ){
            x[i] -= 32;
        }
    }
}
```

7. (1 pont) Írj egy C/C++ vagy Pascal **programot**, amely beolvassa a *matrix.in* állományból a $n \times m$ méretű ($2 \leq n, m \leq 100$), pozitív egészeket tartalmazó mátrixot. Számoljuk meg, hogy hány elemre igaz, hogy egyenlő a szomszédjai (8 irányba) átlagával, majd írassuk ki ezt a számot a képernyőre. A bemenet első sora két egész számot tartalmaz szóközökkel elválasztva (n, m). A következő n sor mindegyike m darab nullánál nagyobb természetes számot tartalmaz, ugyancsak szóközökkel elválasztva.

Példa:

Bemenet:

```
5 5
1 2 2 3 3
2 2 2 3 4
2 2 3 3 4
7 8 7 8 9
1 2 1 7 8
```

Kimenet:

```
3
```

```
float atlag(int i, int j, int x [][102] ){
    int k = 0;
    float s = 0;
    for ( int ii = i-1 ; ii <= i+1 ; ++ii ){
        for ( int jj = j-1 ; jj <= j+1 ; ++jj ){
            if ( ii == i && jj == j || x[ii][jj] == 0 ){
                continue;
            }
            ++k;
            s += x[ii][jj];
        }
    }
    return s/k;
}

int fuggveny ( int n, int m, int x [][102] ){
    int k = 0;
    for ( int i = 1 ; i <= n ; ++i ){
        for ( int j = 1 ; j <= m ; ++j ){
            if ( x[i][j] == atlag(i,j,x) ){
                ++k;
            }
        }
    }
    return k;
}

int main(){
    int n,m;
    int x[102][102];
    ifstream fin("in.txt");
    fin >> n >> m;
    for ( int i = 1 ; i <= n ; ++i ){
```

```
        for ( int j = 1 ; j <= m ; ++j ){
            fin >> x[i][j];
        }
    }
    cout << fuggveny(n,m,x);
    return 0;
}
```

8. (1 pont) Írj Pascal vagy C/C++ **eljárást** (**void-függvényt**), amely paraméterként megkapja az *n* és *m* páratlan természetes számokat ($1 \le n, m \le 100$), valamint egy kétdimenziós tömböt, amely egy *n*×*m* méretű mátrixot tárol (elemei 0-nál nagyobb egészek). Az eljárás írja a ki képernyőre a mátrix „homokórásított” változatát (lásd a példát).

Példa:

Amennyiben *n*=5, *m*=5, és a tömb az alábbi mátrixot tárolja:

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

A képernyőn megjelenítendő módosított mátrix a következő:

```
 1  2  3  4  5
 0  7  8  9  0
 0  0 13  0  0
 0 17 18 19  0
21 22 23 24 25
```

```
void fuggveny ( int n, int m, int x [][100] ){
    for ( int i = 1 ; i < n-1 ; ++i ){
        for ( int j = 0 ; j < m ; ++j ){
            if ( i > j && i+j < n-1 || i < j && i+j > n-1 ){
                x[i][j] = 0;
            }
        }
    }
}
```

9. (1 pont) Az *x*[0..*n*-1][0..*n*-1] tömb tárolta négyzetes mátrix (elemei természetes számok) peremén elhelyezkedő elemekből a következőképpen alakítunk ki párokat, minden *i*=0,*n*-1 esetén:

- a 0. sor *i*. elemét párosítjuk az (*n*-1). oszlop *i*. elemével;
- az (*n*-1). oszlop *i*. elemét párosítjuk az (*n*-1). sor (*n*-*i*). elemével;
- az (*n*-1). sor (*n*-*i*). elemét párosítjuk az 0. oszlop (*n*-*i*). elemével;
- az 0. oszlop (*n*-*i*). elemét párosítjuk a 0. sor *i*. elemével.

Írj Pascal vagy C/C++ **függvényt**, amely paraméterként megkapja az *n* értéket ($1<n<101$) és az *x* kétdimenziós tömböt, és visszatéríti a fenti módon generált párokból adódó legnagyobb összeget.

Példa *n*=3 esetre

```
7 5 9
4 3 2
6 1 8
```

A párokból adódó összegek: 7+9, 5+2, 9+8, 9+8, 2+1, 8+6, 8+6, 1+4, 6+7, 6+7, 4+5, 7+9. A visszatérítendő legnagyobb összeg a 17.

```
int fuggveny ( int n, int x [][100] ){
  int maxOsszeg = x[0][0] + x[0][n-1];
  for ( int i = 0 ; i < n ; ++i ){
        maxOsszeg = max(maxOsszeg,x[0][i]+x[i][n-1]);
        maxOsszeg = max(maxOsszeg,x[i][n-1]+x[n-1][n-1-i]);
        maxOsszeg = max(maxOsszeg,x[n-1][n-1-i]+x[n-1-i][0]);
        maxOsszeg = max(maxOsszeg,x[n-1-i][0]+x[0][i]);
  }
  return maxOsszeg;
}
```