Sapientia Erdélyi Magyar Tudományegyetem
Marosvásárhelyi Kar

Felvételi, 2024. július
– 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 % 2)
| n = n / 10
kiír res
```

Mit ír ki a fenti programrészlet, ha $n = 20240727$?

2

2. (1 pont) Legyen az alábbi pszeudokód programrészlet, ahol a mirror(n) függvény igazat ad vissza, ha az n tükörszám (elölről olvasva ugyanaz, mint hátulról olvasva, pl. 121, 333, 7, 5885), ellenkező esetben hamist:

```
count = 0
res = 0
minden i = 1, n végezd
| ha i % 2 == 0 akkor
| |  ha mirror(a[i]) akkor
| |  | count = count + 1
| |  | res = res + a[i]
| |  | kiír a[i], ’ ’
| |
|
kiír count, ’ ’, res
```

Mit ír ki a fenti programrészlet, ha az a tömb tartalma: 145, 191, 232, 88, 101, 986, 4, 3 ebben a sorrendben, és $n = 8$?

191 88 3 3 282

3. (1 pont) Legyen az alábbi pszeudokód programrészlet:

```
res = 0
amíg n ≠ 0 végezd
| ha (n % 10) == num akkor
| | res = res * 10 + 9 - (n % 10)
| különben
| | res = res * 10 + (n % 10)
| n = n / 10
```

```
kiír res
```

Mit ír ki a fenti programrészlet, ha *n* = 20240727 és *num* = 2?

77704707

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

```
count = 1
┌minden k = 1, n/2 végezd
│ ┌minden j = k, n-k végezd
│ │ a[k][j] = count
│ │ count = count + 2
│ └■
│ ┌minden i = k, n-k végezd
│ │ a[i][n-k+1] = count
│ │ count = count + 2
│ └■
│ ┌minden j = n-k+1, k+1, -1 végezd
│ │ a[n-k+1][j] = count
│ │ count = count + 2
│ └■
│ ┌minden i = n-k+1, k+1, -1 végezd
│ │ a[i][k] = count
│ │ count = count + 2
│ └■
└■
┌ha n % 2 == 1 akkor
│  a[(n+1)/2][(n+1)/2] = count
└■
```

Mi lesz az *a*[1..*n*][1..*n*] kétdimenziós tömb tartalma *n* = 5 majd *n* = 6 esetén?

```
 1  3  5  7  9
31 33 35 37 11
29 47 49 39 13
27 45 43 41 15
25 23 21 19 17
```

```
 1  3  5  7  9 11
39 41 43 45 47 13
37 63 65 67 49 15
35 61 71 69 51 17
33 59 57 55 53 19
31 29 27 25 23 21
```

5. (1 pont) Írj Pascal vagy C/C++ **programot**, amely billentyűzetről bekéri egy személy nemét (egész szám) és korát (egész szám), majd kiírja a képernyőre, hogy a személy jogosult-e nyugdíjra (IGEN/NEM). Ha a személy nő (1) és legalább 62 éves vagy férfi (2) és legalább 65 éves, akkor jogosult nyugdíjra, különben nem.

```
int main(){
    int nem, kor;
    cin >> nem >> kor;
    if( nem == 1 && kor >= 62 || nem == 2 && kor >= 65 ){
        cout << "IGEN";
    }
    else{
        cout << "NEM";
    }
    return 0;
}
```

6. (1 pont) Írj Pascal vagy C/C++ **eljárást** (void-függvényt), amely paraméterként megkapja az *n*

és k természetes számokat (k lehet nagyobb, mint n), valamint egy egydimenziós tömböt, amely egy n-elemű egész számsorozatot tárol. Az eljárás tolja el a tömb elemeit körkörösen jobbra k lépéssel. (Emlékeztető: a maximális pontszám feltétele, hogy a megoldás hatékony legyen idő és tárhely szempontjából)

Példa:

Bemenet:

7 3
1 2 33 4 5 6 71

Kimenet:

5 6 71 1 2 33 4

Figyelem! A fenti példában, ha a k értéke 10 lenne, kimenetként ugyanezt kapnánk, mint $k = 3$ esetén.

```
void reverse(int *arr, int start, int end) {
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

void rotate(int *nums, int n, int k) {
    k = k % n;  // k lehet nagyobb, mint n
    // Fordítsuk meg a teljes sorozatot
    reverse(nums, 0, n - 1);
    // Az első k darab érék megfordítása
    reverse(nums, 0, k - 1);
    // A maradék, n-k érék megfordítása
    reverse(nums, k, n - 1);
}
```

7. **(1 pont)** Írj Pascal vagy C/C++ **programot**, amely beolvas szöveges állományból egy n ($1 \le n \le 52$) természetes számot, majd egy n soros és n oszlopos, természetes számokat tartalmazó mátrix elemeit. A program írja ki a képernyőre a mátrix azon elemeit, amelyek szigorúan kisebbek a közvetlen szomszédjaiknál (olyan elemek, melyek azonos sorban, de szomszédos oszlopban, vagy azonos oszlopban, de szomszédos sorban vannak). A képernyőre írt értékeket szóközzel kell elválasztani (az utolsó kiírt érték után ne legyen szóköz).

Példa:

Bemenet:

4
5 4 7 97
6 **2** 3 4
0 9 8 5
1 3 8 6

Kimenet:

2 0

```
int main(){
    ifstream fin("matrix.txt");
    int n; fin >> n;
    int a[n][n];
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
            fin >> a[i][j];
        }
    }
    bool elso = true;
    for( int i = 0 ; i < n ; ++i ){
        for( int j = 0 ; j < n ; ++j ){
```

```
            if( i>0 && a[i-1][j]<=a[i][j] ||
                j>0 && a[i][j-1]<=a[i][j] ||
                i<n-1 && a[i+1][j]<=a[i][j] ||
                j<n-1 && a[i][j+1]<=a[i][j] ){ continue; }
            if( elso ){ elso = false; }
            else{ cout << " "; }
            cout << a[i][j];
        }
    }
    return 0;
}
```

8. (1 pont) Írj Pascal vagy C/C++ **függvényt**, amely paraméterként megkap egy *n* természetes számot, valamint egy egydimenziós tömböt, amely egy *n*-elemű egész számsorozatot tárol. Egy szöcske az első elemtől indul, és mindig akkora szakaszt ugrik (előre vagy hátra), amennyi a kurrens elem számjegyösszege. Ha a kurrens elem pozitív, akkor előre ugrik, ha viszont negatív, akkor hátra. A függvény térítse vissza, hogy hány ugrásból jut ki a szöcske a tömbből. Amennyiben a szöcske a tömbben rekedne, a függvény a -1 értéket térítse vissza.

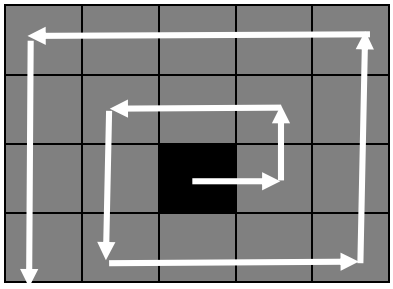
Példa_1:
Bemenet:
5
13 5 -9 77 -22
Kimenet:
-1

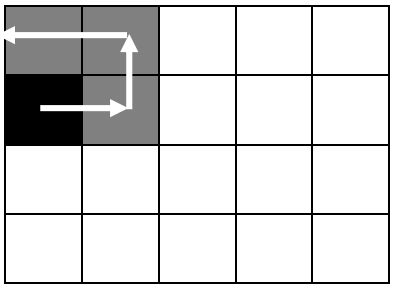
Példa_2:
Bemenet:
5
13 5 -9 77 -21
Kimenet:
3

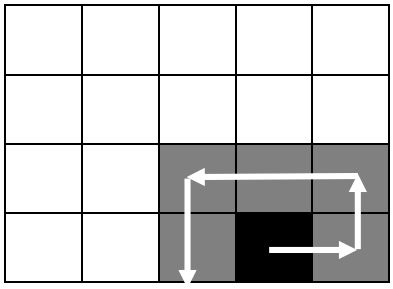
```
int szjossz( int elem ){
    int ossz = 0;
    while( elem ){
        ossz += elem%10;
        elem /= 10;
    }
    return ossz;
}
```

```
int szocske( int n, int *t ){
    int i, lepesszam = 0, volt[n] = {0};
    for( i = 0 ; i>=0 && i<n && !volt[i] ; i += szjossz(t[i]) ){
        volt[i] = 1;
        ++lepesszam;
    }
    if( i>=0 && i<n ) { return -1; }
    else{ return lepesszam; }
}
```

9. (1 pont) Írj Pascal vagy C/C++ **függvényt**, amely paraméterként megkapja az *n*, *m*, *x*, y természetes számokat. Egy csiga egy *n*×*m* méretű mátrix (*x*,*y*) pozíciójáról indul, és, nyilván, csigavonalban halad (lásd az 1. ábrát). A függvény térítse vissza, hogy hány cella érintése után jut ki a csiga a mátrixból. Például, ha a mátrix 4×5 méretű, és a csiga a (3,3) pozícióról indul (fekete cella), akkor a teljes mátrixot bejárja, azaz 20 cellát érint (1. ábra). Viszont, ha a (2,1) pozícióról indulna, akkor 4 cellát érintene (2. ábra). A 3. ábra azt szemlélteti, hogy a (4,4) pozícióról induló csiga 6 cellán halad át mielőtt elhagyja a mátrixot.







```
int csiga1( int n, int m, int x, int y ){
    int db = 1, lepes = 1;
    while( 1 ){
        y += lepes; db += lepes; if( y > m ){ return db-1; }
        x -= lepes; db += lepes; if( x < 1 ){ return db-1; }
        ++lepes;
        y -= lepes; db += lepes; if( y < 1 ){ return db-1; }
        x += lepes; db += lepes; if( x > n ){ return db-1; }
        ++lepes;
    }
}
```

```
int csiga2( int n, int m, int x, int y ){
    int minDistanceToBorder = min(min((x-1),(n-x)),min((y-1),(m-y)));
    int width = 2*minDistanceToBorder + 1;
    int height = 2*minDistanceToBorder + 1;
    if( minDistanceToBorder == m-y ){ ; }
    else if( minDistanceToBorder == x-1 ){ width += 1; }
    else if( minDistanceToBorder == y-1 ){ width += 1; height +=1; }
    else if( minDistanceToBorder == n-x ){ width += 2; height +=1; }
    return width * height;
}
```