INFORMATIKA – tétel
2017

ELMÉLETI TÉTEL:

Sorold fel a rekurzív függvények/eljárások jellemzőit! Szemléltesd a fogalmakat egy konkrét példán! [1 pont]

GYAKORLATI TÉTEL:

1. Legyen az alábbi pszeudokód programrészlet (az `x[1..n]` (n≥2) tömbszakasz egy természetesszám-sorozatot tárol, az `n` változó pedig ennek hosszát):

```
k = 0
minden i = 1,n-1 végezd
  ha x[i] == x[i+1] akkor
    k = 1
  elágazás vége
ciklus vége
kiír k
```

Mit ír ki a program, ha `n=5`, a számsorozat elemei pedig `13`, `17`, `23`, `23`, `53`? [0.5 pont]

1

Adj példát olyan `4`-elemű számsorozatokra, amelyekre `0`, illetve `1` kerül kiírásra! [0.5 pont]

13, 17, 23, 53

13, 17, 23, 23

2. Legyen az alábbi pszeudokód programrészlet (az `x[1..n]` (n≥2) tömbszakasz egy természetesszám-sorozatot tárol, az `n` változó pedig ennek hosszát):

```
k = 0
minden i = 1,n-1 végezd
  minden j = i+1,n végezd
    ha x[i] == x[j] akkor
      k = 1
    elágazás vége
  ciklus vége
ciklus vége
kiír k
```

Mit ír ki a program, ha `n=5`, a számsorozat elemei pedig `13`, `17`, `23`, `3`, `53`? [0.5 pont]

0

Adj példát olyan `4`-elemű számsorozatra, amelyre az első feladat kódrészlete `0`-t a második feladaté pedig `1`-et ír ki! [0.5 pont]

23, 17, 23, 53

3. Legyen az alábbi pszeudokód programrészlet:

```
minden i = 1,3 végezd
  minden j = 1,3 végezd
    minden k = 1,3 végezd
      kiír i,j,k,','
    ciklus vége
  ciklus vége
ciklus vége
```

Mit ír ki a fenti programrészlet? [0.5 pont]

111,112,113,121,122,123,131,132,133,211,212,213,221,222,223,231,232,233,311,312,313,321,322,323,331,

332,333

Hogyan módosítanád a ciklusváltozók határait (aláhúzott értékek), hogy az `{1,2,3,4}` halmaz harmadrendű kombinációi (`123,124,134,234`) kerüljenek kiírásra (hatékony módon)? [0.5 pont]

```
┌minden i = 1,2 végezd
|  ┌minden j = i+1,3 végezd
|  |  ┌minden k = j+1,4 végezd
|  |  |  kiír i,j,k,','
|  |  └■
|  └■
└■
```

4. Legyen az alábbi pszeudokód nyelven írt rekurzív eljárás, ahol `x` egydimenziós tömb `1`-től `n`-ig indexelt cellákkal (`x[1..n]`) (az `x` és `n` aláhúzott paraméterek cím szerint, `k` pedig érték szerint kerül átadásra). Mi kerül kiírásra az `E(x,3,1)` eljáráshívás nyomán, ha tudjuk, hogy

- az `f` függvény aszerint térít vissza IGAZ-at/ HAMIS-at, hogy az `x[k]` elem különbözik-e az `x[1..k-1]` szakasz elemeitől vagy sem;
- a `kiír(x,n)` eljárás kiírja az `x[1..n]` tömbszakasz elemeit, majd újsorba ugratja a kurzort.

[1 pont]

```
┌E(x[],n,k)
|  ┌minden x[k] = 1,n végezd
|  |  ┌ha f(x,k) akkor
|  |  |  ┌ha k < n akkor
|  |  |  |  E(x,n,k+1)
|  |  |  | különben
|  |  |  |  kiír(x,n)
|  |  |  └■
|  |  └■
|  └■
└■
```

123
132
213
231
312
321

Megjegyzés: Az 5-9 feladatok esetében, használj alprogramot valahányszor célszerűnek találod. Törekedj hatékony megoldásra mind az időigény, mind a tárhely szempontjából. Lásd el beszédes kommentekkel programjaidat.

5. Írj Pascal vagy C/C++ programot, amely billentyűzetről beolvassa az `n` értéket (`2≤n≤999`), valamint egy `n` elemű növekvő számsorozatot (elemei egész számok), majd kiír a képernyőre egy megfelelő üzenetet aszerint, hogy a számsorozat halmaznak tekinthető-e (elemei páronként különböznek). [1 pont]

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
// az összehasonlítás pillanatában
// az a és b változók az utolsóelőttiként és utolsóként beolvasott számokat tárolják
        int n, a, b;
        printf("n: "); scanf("%i", &n);
        printf("szam: "); scanf("%i", &a);
        while (n > 1){
                printf("szam: "); scanf("%i", &b);
                if (b == a) {
                        printf("nem hamaz\n");
                        return 0;
                }
                a = b;
                n--;
        }
        printf("halmaz\n");
```

```
        return 0;
}
```

6. Adottak n síkbeli pont x és y koordinátái. Írj Pascal vagy C/C++ alprogramot, amely paraméterként megkapja az n értéket (2≤n≤999), valamint a pontok koordinátáit (valós értékek), és kiírja a képernyőre a két legközelebbi pont koordinátáit. [1 pont]

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

//a beszédes plusz kiírások nem jelentettek se több, se kevesebb pontot

void szamol(int n, float *x, float *y){
 int i, j, temp1, temp2;
 float d, maxd = LONG_MAX;
 int maxi = 0, maxj = 0;
 for(i = 0; i < n - 1; i++){
  for(j = i + 1; j < n; j++){
   temp1 = x[i] - y[i];
   temp2 = x[j] - y[j];
   d = sqrt(temp1 * temp1 + temp2 * temp2);
   printf("%5.2f %5.2f %5.2f %5.2f %5.2f \n", d, x[i], y[i], x[j], y[j]);
   if( d < maxd){
    maxd = d; maxi = i; maxj = j;
   }
  }
  printf("A legkisebb tavolsag\n");
  printf("%5.2f %5.2f %5.2f %5.2f %5.2f ", maxd, x[maxi], y[maxi], x[maxj], y[maxj]);
 }
}
int main(){ //nem volt kötelező a main-t is megírni
 float *a, *b;
 int n, i;
 FILE *fin;
 fin = fopen("be.txt","r");
 fscanf(fin,"%i",&n);
 a = (float*)malloc(n * sizeof(float));
 b = (float*)malloc(n * sizeof(float));
 for(i = 0; i < n; i++){
  fscanf(fin, "%f", &a[i]);
 }
 for(i = 0; i < n; i++){
  fscanf(fin, "%f", &b[i]);
 }
 szamol(n, a, b);
 return 0;
}
```

7. Írj Pascal vagy C/C++ programot, amely a matrix.txt állományból beolvassa az n és m értékeket (2≤n,m≤999), valamint egy nxm méretű mátrixot (elemei egész számok a [0,10000) intervallumból), amelynek minden sora egy-egy halmaznak tekintendő (elemeik páronként különböznek). Írasd ki az eredmeny.txt állományba a mátrix sorai képezte halmazok metszetét (azaz, azok közös elemeit). [1 pont]

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

using namespace std;

///
///  Mivel ismerjuk az intervallumot, amelybol a szamok szarmazhatnak,
///  hasznalhatunk karakterisztikus tombot.
///

int main(){
    int n, m, x, mmax = 0; /// mmax - a legnagyobb elofordulo ertek
    int kar[10000] = {0};
/// a legnagyobb elem, ami elofordulhat az a 9999, lenullazzuk a tomb elemeit.
```

```
    ifstream f("be.txt");
    if (f.fail()){
        cout << "Allomanymegnyitasi hiba!";
        return 0;
    }

    ofstream g("ki.txt");
    if (f.fail()){
        cout << "Allomanymegnyitasi hiba!";
        return 0;
    }

    f >> n >> m;

    /// A matrix beolvasasa szukseges, de az elemeket nem szukseges eltarolni,
    /// menet kozben fel is dolgozzuk oket.
    for (int i=0; i<n; ++i)
        for (int j=0; j<m; ++j){
            f >> x;
            kar[x]++;
            if (mmax < x)
                mmax = x;
        }
    f.close();

    /// Minden olyan eleme a karakterisztikus tombnek, ami n-szer fordult elo,
    /// definicio szerint minden sorban elofordult egyszer, tehat eleme a metszetnek
    for (int i=0; i<=mmax; ++i){
        if (kar[i] == n)
            g << setw(5) << i;
    }

    return 0;
}
```

8. Írj Pascal vagy C/C++ kódrészletet, amely kiírja a képernyőre az `a[1..n][1..m][1..p]` háromdimenziós tömb sorai képezte halmazok (elemeik páronként különböznek) egyesített halmazát (a tömb első sora elemeit az `a[1][1][1..p]` cellák, az utolsó sor elemeit pedig az `a[n][m][1..p]` cellák tárolják). ($2 \le n,m,p \le 99$; a tömb elemei egész számok) [1 pont]

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <algorithm>

using namespace std;

///
///  Mivel nem ismerjuk az intervallumot, amelybol a szamok szarmazhatnak,
///  nem hasznalhatunk karakterisztikus tombot!
///  Ehelyett novekvo sorba rendezzuk az osszes elemet, es kiirjuk egy bejarassal
///  az egymastol kulonbozoket.
///

///3D tomb kiiratasahoz hasznalhato fuggveny, "laponkent" ir ki
void kiir3D(int ***t, int n, int m, int p);
///1D tomb kiiratasahoz hasznalhato fuggveny
void kiir1D(int *t, int n);

/*****************************************************************************
      Az egyesitett halmaz elemeit kiiro fuggveny, ami a 8. feladat egy lehetseges
hatekony megoldasa.
*****************************************************************************/
void egyesit(int ***a, int n, int m, int p){
    int nn = n*m*p;
    ///A haromdimenzios tombot atalakitjuk egydimenzios tombbe
    int *t = new int [nn];
    for (int i=0; i<n; ++i)
```

```
        for (int j=0; j<m; j++)
            for (int k=0; k<p; k++)
                t[i*m*p+j*p+k] = a[i][j][k];
    cout << "Egdimenzios formaban: " << endl;
    kiir1D(t,nn);

    ///Elrendezzuk a tombot a beepitett QuickSort-tal (sort)
    sort(t,t+nn);
    cout << "Rendezes utan: " << endl;
    kiir1D(t,nn);

    ///Kiirjuk az egymastol kulonbozo elemeket - minden elemvaltaskor kiirjuk az uj
    ///elemet
    cout << "Az egyesitett halmaz elemei: " << endl << setw(4) << t[0];
    ///beallitjuk az uj elemet mar kiirtnak
    int elozo = t[0];
    for (int i=1; i<nn; ++i){
        ///ha elemvaltas tortent
        if (t[i] != elozo){
            cout << setw(4) << t[i];
            ///beallitjuk az uj elemet mar kiirtnak
            elozo = t[i];
        }
    }
}
/*****************************************************************************
      A kodreszlet vege.
*****************************************************************************/

int main(){
    int ***t, n, m, p;

    ifstream f("be.txt");
    if (f.fail()){
        cout << "Allomanymegnyitasi hiba!";
        return 0;
    }

    f >> n >> m >> p;

    /// Dinamikusan lefoglalunk helyet egy adott meretu 3D tombnek
    /// Alternativ lehetoseg, de nem mindig mukodik tarhelyproblemak miatt:
    ///       int t[n][m][p];

    t = new int** [n];
    for (int i=0; i<n; ++i){
        t[i] = new int * [m];
        for (int j=0; j<m; ++j)
            t[i][j] = new int [p];
    }

    ///Beolvassuk a szamokat
    for (int i=0; i<n; ++i)
        for (int j=0; j<m; j++)
            for (int k=0; k<p; k++)
                f >> t[i][j][k];
    f.close();

    cout << "A beolvasott elemek: " << endl;
    kiir3D(t,n,m,p);

    /// Vegrehajtjuk az egyesitett halmazhoz szukseges muveleteket
    egyesit(t,n,m,p);

    return 0;
}

///Az egyesitett halmaz elemeit kiiro fuggveny
///3D tomb kiiratasahoz hasznalhato fuggveny, "laponkent" ir ki
void kiir3D(int ***t, int n, int m, int p){
```

```
    for (int i=0; i<n; ++i){
        for (int j=0; j<m; j++){
            for (int k=0; k<p; k++)
                cout << setw(4) << t[i][j][k];
            cout << endl;
        }
        cout << endl << endl;
    }
}

///1D tomb kiiratasahoz hasznalhato fuggveny
void kiir1D(int *t, int n){
    for (int k=0; k<n; k++)
        cout << setw(4) << t[k];
    cout << endl << endl;
}
```

9. Egy felhőkarcoló földszintjén, egy egyszemélyes felvonó előtt, `n` személy áll. A felvonó sajátossága, hogy (1) minden használat után automatikusan visszatér a földszintre, és (2) az `i`-edik emeletre éppen `i` időegység alatt megy fel és tér vissza. Ha a be- és kiszállási időt elhanyagoljuk, akkor mennyi a *földszinti* minimális ÖSSZVÁRAKOZÁSI (a személyek várakozási ideinek összege) idő (hány időegység) ahhoz, hogy az összes személy hazajusson? Írj Pascal vagy C/C++ programot, amely szöveges állományból beolvassa az `n` értéket (`1≤n≤999`), majd pedig azt, hogy a személyek mely emeleteken laknak (nullánál nagyobb természetes számok), és kiírja a képernyőre a kért minimális időegységszámot. [1 pont]

Példa bement:

```
5
3 9 2 14 7
```

Példa kimenet:

```
40
```

Ötlet (feltételezzük, hogy az emeletértékeket az e[1..n] tömb tárolja):

- Mivel az elsőkét liftező személy „utazási ideje” (n-1) személy várakozási idejébe kerül bele, ezért a legalacsonyabb emeleten lakó személynek kell elsőként használnia a felvonót… Ha a legmagasabb emeleten lakó személy használja utolsóként a felvonót, akkor az ő „utazási ideje” egyetlen személy várakozási idejébe se kerül bele.
- Rendezzük az *e* tömb elemeit növekvő sorrendbe (qsort)
- Az eredmény: $\sum_{i=1}^{n-1}(e[i]*(n-i))$