BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA-INFORMATIKA KAR

MATEK-INFO VERSENY – 2017, április 1
INFORMATIKA
I. TÉTELSOR

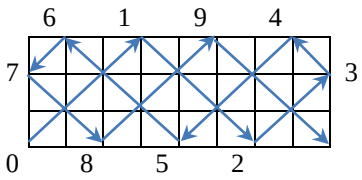
A versenyzők figyelmébe:

1. A feladatok megoldásait *pszeudokódban* vagy egy *programozási nyelvben* (*Pascal/C/C++*) kell megadnotok.
2. A megoldások értékelésekor az első szempont az algoritmus ***helyessége***, majd a ***hatékonysága***, ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti.
3. A tulajdonképpeni megoldások előtt, ***kötelezően leírjátok szavakkal az alprogramokat, és megindokoljátok a megoldásotok lépéseit***. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket), amelyek segítik az adott megoldás technikai részleteinek megértését. Adjátok meg az azonosítók jelentését és a fölhasznált adatszerkezeteket. Ha ez hiányzik, a tételre kapható pontszámotok 10%-kal csökken.
4. Ne használjatok különleges fejállományokat, előredefiniált függvényeket (például STL, karakterláncokat feldolgozó sajátos függvények stb.).

I. Tétel (35 pont)

1. Fénysugár (20 pont)

Legyen egy tükrökből kialakított, téglalap alakú keret. Egy fénysugár elindul a téglalap bal alsó sarkából, 45^o fokos szöget alkotva a téglalap alsó oldalával, és nekiütközik a téglalap felső vagy jobboldali oldalának. Itt tükröződik (elindul egy másik oldal felé, szintén 45^o fokos szöget alkotva azzal az oldallal, amelybe beleütközött). Így folytatja az útját, amíg a keret valamelyik sarkába nem ér.



Írjatok alprogramot, amely kiszámítja, hogy hányszor (***váltSz***) változtatja a tükröződés irányát a fénysugár, amíg leáll valamelyik sarokban. A kiindulási pontot nem számítjuk be ebbe a számba. Az alprogram bemeneti paraméterei a téglalap hossza ($1 < \boldsymbol{a} < 10\,000$) és szélessége ($1 < \boldsymbol{b} < 10\,000$), míg a ***váltSz*** kimeneti paraméter.
1. *Példa:* ha $\boldsymbol{a} = 8$ és $\boldsymbol{b} = 3$, akkor ***váltSz*** = 9.
2. *Példa:* ha $\boldsymbol{a} = 8$ és $\boldsymbol{b} = 4$, akkor ***váltSz*** = 1.

2. „Erős” számok (15 pont)

Egy nullától különböző ***sz*** természetes számnak az *erőssége* ***k***, ha bináris alakjában pontosan ***k*** darab 1-es számjegy található. Például, a 23 erőssége 4 (kettes számrendszerben felírva, 4 darab 1-es számjegye van). Adott számsorozat ***k*** *erősségű csoport*jának nevezzük azt a részsorozatot, amely a sorozat ***k*** erősségű elemeit tartalmazza, az elemek eredeti sorrendjében. Például, az ***s*** = (7, 12, 3, 13, 24, 19), sorozat ***k*** = 2 *erősségű csoportja* a (12, 3, 24) részsorozat.

Írjatok alprogramot, amely meghatároz minden *erősségi csoportot*, amelyek az adott ***x*** sorozat elemeiből létrehozhatók. Bemeneti paraméterek: a nem nulla, 30 000-nél kisebb, különböző természetes számokból álló ***x*** sorozat és a sorozat ***n*** hossza ($1 < \boldsymbol{n} < 100$). Kimeneti paraméterek: a ***csSzáma*** (a csoportok száma) és a ***csoportok*** (a létrehozott csoportok, erősségük szerint növekvően rendezve).
Példa: ha $\boldsymbol{n} = 6$ és $\boldsymbol{x} = (12, 3, 24, 16, 15, 32)$, akkor ***csSzáma*** = 3, és ***csoportok***: (16, 32), (12, 3, 24), (15).

II. Tétel (15 pont)

Adott a következő alprogram, amelynek három, természetes szám bemeneti paramétere van: ***a***, ***b*** és ***sz***, amelyeknek értékei kisebbek, mint 10 000:

```
Algoritmus f(a, b, sz):
   k ← 0
   Amíg b < sz végezd el:
      k ← k + 1
      b ← a + b
      a ← b - a
   vége(amíg)
   térítsd k
Vége(algoritmus)
```

a. Adjátok meg a feladat szövegét, amelyet ez az algoritmus old meg, ha ***a*** = 1 és ***b*** = 0 értékekre hívjuk meg.

b. Mit térít az **`F(1,0,10)`** hívás?

c. Írjátok le egy *rekurzív* változatát a fenti algoritmusnak, megőrizve az iteratív (nem rekurzív) változat fejlécét.

III. Tétel (40 pont)

Előszelet

Sors-számjegynek hívjuk azt a természetes számot, amelyet adott természetes számra a következőképpen számítunk ki: összeadjuk a szám számjegyeit, majd a kapott összeg számjegyeit, és így tovább, amíg a kapott összeg nem válik egyszámjegyű számmá. Például, a 182 *sors-számjegye* 2 (1 + 8 + 2 = 11, 1 + 1 = 2).

Egy pontosan ***k*** számjegyű ***p*** számot egy legkevesebb ***k*** számjegyű ***q*** szám *előszeleté*nek nevezünk, ha a ***q*** szám első ***k*** számjegyéből alkotott szám (balról jobbra tekintve) egyenlő ***p***-vel. Például, 17 előszelete 174-nek, és 1713 előszelete 1 713 242-nek.

Legyen az ***sz*** természetes szám ($0 < sz \leq 30\,000$) és az ***m*** sorral és ***n*** oszloppal ($0 < m \leq 100$, $0 < n \leq 100$) rendelkező ***A*** mátrix (kétdimenziós tömb), amelynek elemei 30 000-nél kisebb természetes számok. Írjatok programot, amely meghatározza és kiírja az ***sz*** szám *leghosszabb előszeletét*, amelyet az adott mátrix elemeinek megfelelő *sors-számjegyeiből* fel lehet építeni. Egy ilyen sors-számjegyet akárhányszor fel lehet használni. Ha nem építhető fel előszelet, a program írja ki a „*nem létezik előszelet*” üzenetet.

Példa*:** ha ***sz = 12319, ***m*** = 3, ***n*** = 4 és a mátrix:

$$A = \begin{pmatrix} 182 & 12 & 274 & 22 \\ 22 & 1 & 98 & 56 \\ 5 & 301 & 51 & 94 \end{pmatrix},$$

akkor a leghosszabb előszelet 1231, a megfelelő sors-számjegyek pedig:

Mátrixelem értéke	182	12	274	22	1	98	56	5	301	51	94
Sors-számjegy	2	3	4	4	1	8	2	5	4	6	4

A megoldásban fölhasználjátok a következő alprogramokat:

a. a bemeneti adatok beolvasása billentyűzetről;

b. adott számhoz rendelhető sors-számjegy meghatározása;

c. a leghosszabb előszelet meghatározása;

d. a leghosszabb előszelet kiírása a képernyőre; ha nincs előszelet, a megfelelő üzenet kiírása.

Megjegyzések:

1. Minden tétel kidolgozása kötelező.
2. A megoldásokat a vizsgalapokra írjátok, (a piszkozatokat nem vesszük figyelembe).
3. Hivatalból jár 10 pont.
4. Rendelkezésetekre áll 3 óra.

BAREM – VARIANTA I

OFICIU ..... 10 puncte

SUBIECTUL I ..... 35 puncte

1. Rază ..... 20 puncte

Varianta 1: determinarea corectă a valorii ***nrSchimb*** bazată pe utilizarea *cmmdc*(***a***, ***b***) ..... 20 puncte

- *cmmdc*(***a***, ***b***) (sau *cmmmc*(***a***,***b***)) ..... 5 puncte
- calculul valorii ***nrSchimb*** ..... 15 puncte

Varianta 2: determinarea corectă a valorii ***nrSchimb*** cu alt algoritm corect (simulare) ..... 15 puncte

2. Numere cu „forță” ..... 15 puncte

- stabilirea forței unui număr ..... 5 puncte
- determinarea numărului grupurilor de elemente cu aceeași forță (***nrGr***) și a componenței lor (***grupuri***) ..... 10 puncte

SUBIECTUL II ..... 15 puncte

Numărul numerelor *Fibonacci* mai mici decât numărul *nr* dat

Cerința a.

- enunț problemă ..... 5 puncte

Cerința b.

- rezultat calculat corect (7) ..... 3 puncte

Cerința c.

- algoritm ..... 3 puncte
- autoapel corect ..... 2 puncte
- condiție de oprire din recursivitate ..... 2 puncte

SUBIECTUL III ..... 40 puncte

Prefix

Subprograme:

- citirea datelor de intrare ..... 3 puncte
- determinarea cifrei de control asociată unui număr ..... 5 puncte
- determinarea celui mai lung prefix ..... 15 puncte
- afișarea celui mai lung prefix/mesaj ..... 3 puncte

Program principal: ..... 3 puncte

- comunicare prin parametri: (signatura subalgorimilor și apelul corect) ..... 5 puncte
- lizibilitate:
  - comentarii ..... 2 puncte
  - indentare ..... 2 puncte
  - denumiri sugestive ..... 2 puncte

UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Concurs Mate-Info - 1 aprilie 2017

Proba scrisă la Informatică
VARIANTA I

```
// SUBIECTUL I.1

//determina cmmdc a 2 numere a si b
int cmmdc(int a, int b){
	if ((a == b) && (a == 0)){
		return 1;
	}
	if (a * b == 0){
		return a + b;
	}
	while (a != b){
		if (a > b)
			a -= b;
		else
			b -= a;
	}
	return a;
}

// calcularea numărului de schimbări de direcție a razei
int raza(int a, int b){
	int nrSchimb;
	int d = cmmdc(a, b);
	nrSchimb = b / d + a / d - 2;
	return nrSchimb;
}

// SUBIECTUL I.2

const int MAXSIZE = 100;
const int MAXCIF = 32;

// calcularea fortei unui numar
int calculForta(int nr){
	int forta = 0;
	do{
		nr &= nr - 1; // „și” logic intre nr si nr-1
		forta++;
	} while (nr);
	return forta;
}

//gruparea numerelor pe clase de forta
//grupurile se vor memora intr-un tablou bidimensional
//fiecare linie i corespunde grupului de forta i
//primul element de pe linia i reprezinta dimensiunea grupului de forta i
//urmatoarele elemente sunt numerele din sirul x cu forta i
void forte(int n, int x[], int &nrGr, int grupuri[][MAXSIZE]){
	for (int i = 1; i < MAXCIF; i++)  // inițializarea tabloului grupuri
		grupuri[i][0] = 0;
	nrGr = 0;
	for (int i = 0; i < n; i++){
		int forta = calculForta(x[i]);    // determinarea forței elementului x[i]
		if (grupuri[forta][0] == 0)
			nrGr++;
		int pos = grupuri[forta][0] + 1;
		grupuri[forta][pos] = x[i];
		grupuri[forta][0]++;
	}

}
```

```
// SUBIECTUL II
int fRec(int a, int b, int nr){
	if (b < nr)
		return fRec(b, a + b, nr) + 1; // (*)
	else
		return 0;
}



// SUBIECTUL III
const int MAX = 100;
const int NRMAXCIFRE = 10;

//citirea unui numar
void citireNumar(int &nr){
	cin >> nr;
}

//citirea elementelor unei matrici
void citireMatrice(int &m, int &n, int A[][MAX]){
	cin >> m >> n;
	for (int i = 0; i < m; i++){
		for (int j = 0; j < n; j++)
			cin >> A[i][j];
	}
}

//citirea numarului si a elementelor din matrice
void citireDate(int &nr, int &m, int &n, int A[][MAX]){
	citireNumar(nr);
	citireMatrice(m, n, A);
}

//calcularea cifrei de control a unui numar
int cifraControl(int x){
	while (x > 9){
		int y = x;
		int s = 0;
		while (y > 0){
			s += y % 10;
			y /= 10;
		}
		x = s;
	}
	return x;
}
```

```
//determinarea prefixului maxim
//matricea se parcurgere o singură dată și se construiește un vector de apariții
//pentru cifrele de control corespunzătoare elementelor din matrice.
//Se reține într - un vector cifrele numărului dat(nr).
//Se parcurge vectorul acestor cifre(începând cu cifra cea mai semnificativă) și
//se verifică apariția cifrelor în vectorul de apariții construit anterior
int prefixMaxCifre(int nr, int m, int n, int A[][MAX], int cifre[], int &nrCifre){
	bool aparitii[NRMAXCIFRE];
	int i = 0;
	for (i = 0; i < 10; i++)    // initializarea vectorului de frecvente
		aparitii[i] = 0;
	for (i = 0; i < m; i++){
		for (int j = 0; j < n; j++){
			aparitii[cifraControl(A[i][j])] = 1;     //  aparitia cifrei de control
		}                                    // corespunzatoare elementelui din matricea A
	}
	nrCifre = 0;         // numarul cifrelor numarului nr dat
	while (nr > 0){
		cifre[nrCifre++] = nr % 10; // elementele sirului cifrelor numarului nr dat
		nr /= 10;
	}

	i = nrCifre - 1;     // parcurgem sirul cifrelor
	while ((i >= 0) && (aparitii[cifre[i]])) // cifra de control = cu cifra curenta din nr
		i--;
	return nrCifre - i - 1;
}


//se afiseaza prefixul maxim de lungime lung cu cifre din vectorul de cifre
void afisarePrefix(int lung, int cifre[], int nrCifre){
	if (lung == 0)
		cout << "nu exista prefix";
	for (int i = 0; (i < lung && i < nrCifre); i++)
		cout << cifre[nrCifre - i - 1];
	cout << endl;
}



int main(){

	int nr = -1;
	int m = -1;
	int n = -1;
	int A[MAX][MAX];
	citireDate(nr, m, n, A);

	int cifre[NRMAXCIFRE];
	int nrCifre = 0;
	int lung = prefixMaxCifre(nr, m, n, A, cifre, nrCifre);
	afisarePrefix(lung, cifre, nrCifre);

	return 0;
}
```