Conversii

Capitolul 5

- **Sisteme de numerație**
- **Conversii între două baze de numerație**
- **Operații cu numere naturale în diverse baze de numerație**
- **Criterii de divizibilitate în diverse baze de numerație**
- **Implementări sugerate**
- **Probleme propuse**
- **Soluțiile problemelor**

5.1. Sisteme de numerație

Prin ***sistem de numerație*** înțelegem ansamblul regulilor de grupare a elementelor unei mulțimi finite cu scopul numărării lor şi a regulilor de reprezentare simbolică a numărului obținut.

Simbolurile (semnele grafice) cu ajutorul cărora se reprezintă numerele se numesc *cifre*.

În funcție de modul de grupare şi de ordonare a semnelor care se folosesc pentru reprezentarea simbolică a numerelor, deosebim ***sistemele aditive*** de ***sistemele poziționale***.

Printre sistemele de numerație *aditive* se numără sistemul de numerație *egiptean* şi sistemul de numerație *roman*.

Sistemul de numerație roman foloseşte şapte semne distincte, numite *cifre romane*.

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Dintre cele şapte, cifrele **`I`**, **`X`**, **`C`** şi **`M`** nu se pot repeta (scriind alăturat) de mai mult de trei ori în descrierea unui număr.

Inițial sistemul de formare a numerelor romane era bazat strict pe adunare. Cifrele unui număr se aşezau în ordine descrescătoare a valorilor, iar valoarea numărului se calcula prin însumarea tuturor valorilor cifrelor.

Exemplu

`MDXXI` = 1000 + 500 + 10 + 10 + 1 = 1521

Astfel, unele numere aveau o reprezentare foarte lungă şi pentru a le scurta, s-a introdus varianta de reprezentare prin diferenţă. Dacă una din cifrele **`I`**, **`X`** sau **`C`** se află înaintea unei cifre cu valoare mai mare, atunci valoarea cifrei respective va fi scăzută din valoarea cifrei mai mari.

Exemple
`IV` = 5 – 1 = 4; **`IX`** = 10 – 1 = 9; **`XL`** = 50 – 10 = 40

Aceste forme de reprezentare sunt supuse câtorva restricţii, care asigură unicitatea reprezentării unui număr. De exemplu:
1. Cifrele **`I`**, **`X`**, **`C`** sau **`M`** nu se pot repeta pe mai mult de trei poziţii consecutive;
2. Cifrele **`V`**, **`L`** şi **`D`** nu se pot repeta;
3. Numai cifrele **`I`**, **`X`**, şi **`C`** pot fi scăzute;
4. Numai valoarea unei singure cifre poate fi scăzută (de exemplu numărul **`IIIV`** nu există);
5. Cifra scăzută trebuie să aibă o valoare de *cel puţin* o zecime din cifra din care se scade (de exemplu numărul **`ID`** nu există).

Cel mai mare număr pe care îl putem reprezenta, folosind aceste simboluri şi respectând regulile de mai sus este: **`MMMCMXCIX`** = 3999. Datorită faptului că numerele romane sunt foarte lungi, operaţiile cu numere astfel reprezentate se efectuează greoi.

Observăm că o cifră din cadrul unui astfel de număr *are aceeaşi valoare absolută indiferent de poziţia pe care o ocupă*.

Cel mai răspândit sistem de numeraţie în zilele noastre este ***sistemul zecimal*** care utilizează zece simboluri: 0,1, 2, 3, 4, 5, 6, 7, 8, 9, numite *cifre arabe*.

Numărul simbolurilor reprezintă *baza sistemului de numeraţie*.

Sistemul zecimal este *un sistem de numeraţie poziţional*, adică *poziţia* unei cifre în număr indică *rangul* acesteia. Valoarea rangului precizează cu ce putere a lui 10 se va înmulţi cifra pentru a determina valoarea numărului.

Pentru a nu se produce confuzii între două numere scrise în sisteme de numeraţie poziţionale diferite, s-a convenit ca fiecărui număr să i se ataşeze un indice corespunzător bazei de numeraţie în care este scris.

Exemplu
$273_{10} = 2 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$ (se va citi: două **sute** şapte **zeci** şi trei)

Orice număr natural n poate fi scris în sistemul zecimal în mod unic în forma:
$n = a_n a_{n-1} \ldots a_0$, unde a_i (i = 1, 2, ..., n) sunt cifre din mulţimea {0, 1, 2, …, 9}, iar valoarea lui n este dată de suma: $a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + 10^0 \cdot a_0$.

Pentru a defini un sistem de numerație se descrie mulțimea de simboluri folosită în reprezentare, regulile de grupare a simbolurilor în număr şi se prezintă operațiile definite pe mulțimea numerelor reprezentate în sistemul de numerație respectiv.

În sistemul de numerație zecimal, pentru a număra obiectele, acestea se grupează câte 10. Numărul grupurilor reprezintă de câte ori se cuprinde 10 în număr. Acestea, grupate din nou câte zece ne dau numărul sutelor etc. În mod similar, într-un sistem pozițional având baza *b*, obiectele se vor grupa câte *b*. Regulile de reprezentare a unui număr în orice sistem de numerație pozițional sunt similare regulilor care funcționează în sistemul zecimal.

Sistemul de numerație binar este caracterizat de:

- Baza sistemului de numerație este 2.
- Mulțimea de simboluri care stă la baza sistemului este {0, 1}.

De exemplu, numărul 1110_2 reprezintă valoarea $1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 13_{10}$.

Sistemul de numerație octal are următoarele caracteristici:

- Baza sistemului este 8.
- Mulțimea de simboluri care stă la baza sistemului este {0, 1, 2, 3, 4, 5, 6, 7}.

De exemplu, numărul 2673_8 reprezintă valoarea $2 \cdot 8^3 + 6 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 = 1467_{10}$.

Un alt sistem de numerație frecvent utilizat în lumea calculatoarelor este ***sistemul de numerație hexazecimal***. Caracteristicile acestuia sunt:

- Baza sistemului este 16.
- Valorile cu care lucrează acest sistem de numerație sunt numerele de la 0 la 15. Deoarece numerele mai mari decât 9 sunt combinații de simboluri (spre exemplu 10 se formează din 1 şi din 0) ele nu pot fi utilizate la rândul lor ca simboluri. Există însă o convenție prin care se foloseşte litera A pentru valoarea 10, litera B pentru valoarea 11 şi aşa mai departe. În aceste condiții simbolurile care stau la baza acestui sistem de numerație sunt 0, 1, ..., 9, A, B, C, D, E şi F.

5.2. Conversii între două baze de numerație

5.2.1. Conversia unui număr dintr-o bază dată în baza 10

A. Algoritm de conversie pentru numere întregi
Efectuând calculele cerute de descompunerea după puterile bazei, putem obține în cazul oricărui sistem de numerație valoare numărului dat în baza 10.

Exemplu
$2673_8 = 2 \cdot 8^3 + 6 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 =$
$= 2 \cdot 512 + 6 \cdot 64 + 7 \cdot 8 + 3 \cdot 1 =$
$= 1024 + 384 + 56 + 3 =$
$= 1467_{10}$

Tragem concluzia că pentru a obține reprezentarea în baza 10 a unui număr natural reprezentat într-o bază *b*, trebuie să descompunem numărul după puterile bazei şi să efectuăm calculele în baza 10.

B. Algoritm de conversie pentru partea fracționară a unui număr
Pentru a converti un număr fracționar dintr-o bază *b* în baza 10 aplicăm acelaşi algoritm ca şi pentru partea întreagă: dezvoltăm numărul cifră cu cifră după puterile bazei şi apoi efectuăm calculele în baza 10.

Exemplu

$$0{,}24_5 = 2 \cdot \frac{1}{5^1} + 4 \cdot \frac{1}{5^2} = 0{,}56_{10}$$

5.2.2. Conversia din baza 10 într-o altă bază de numerație

A. Algoritm de conversie pentru numere întregi
Conversia unui număr natural reprezentat în baza 10 într-o altă bază de numerație *b* se face prin împărțiri succesive la *b*, valoarea numărului fiind înlocuită la fiecare pas prin câtul împărțirii. Resturile scrise în ordine inversă reprezintă cifrele numărului căutat.

Exemplu
$1467 = 8 \cdot 183 + \mathbf{3}$
$183 = 8 \cdot 22 + \mathbf{7}$
$22 = 8 \cdot 2 + \mathbf{6}$
$2 = 8 \cdot 0 + \mathbf{2}$
deci $1467_{10} = 2673_8$

B. Algoritm de conversie pentru partea fracționară a unui număr
Conversia unui număr fracționar reprezentat în baza 10 într-o bază *b* se face prin înmulțiri succesive cu *b*. La fiecare pas, partea întreagă a rezultatului înmulțirii ne dă câte o cifră din rezultat, înmulțirea continuând cu partea fracționară a rezultatului. Algoritmul va avea atâția paşi câte cifre zecimale dorim să determinăm.

Exemplu

Vom converti numărul $0{,}23_{10}$ din baza 10 în baza 5, cu trei zecimale.

Număr în baza 10	Calcul	Produs	Nu măr în baza 5
$0{,}23_{10}$	$0{,}23_{10} \cdot 5$	**1**,15	
$0{,}15_{10}$	$0{,}15_{10} \cdot 5$	**0**,75	
$0{,}75_{10}$	$0{,}75_{10} \cdot 5$	**3**,75	$0{,}\mathbf{103}_5$

5.2.3. Algoritm de conversie din baza 2 într-o bază de forma 2^k şi invers

Pentru a arăta corespondența dintre numerele scrise în sistemul binar şi simbolurile sistemelor de numerație cu baza 2^k, prezentăm următorul tabel:

valoare zecimală	***k* = 1**	***k* = 2**	***k* = 3**	***k* = 4**
	baza 2	**baza 4**	**baza 8**	**baza 16**
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	10	4	4
5	101	11	5	5
6	110	12	6	6
7	111	13	7	7
8	1000	20	10	8
9	1001	21	11	9
10	1010	22	12	A
11	1011	23	13	B
12	1100	30	14	C
13	1101	31	15	D
14	1110	32	16	E
15	1111	33	17	F

A. Algoritm de conversie din baza 2 în baza 2^k
Pentru a converti, de exemplu, numărul $10111{,}01_2$ în baza 8 (= 2^3), se completează numărul cu 0-uri nesemnificative în stânga şi în dreapta până când avem grupuri complete de câte trei cifre. Grupurile de cifre se formează pornind de la virgula zecimală spre stânga, respectiv spre dreapta. Numărul devine 0**10 111 , 01**0. La pasul următor înlocuim fiecare grup de câte trei cifre binare cu simbolul bazei 8 corespunzătoare grupului (vezi valorile din tabel) şi obţinem numărul convertit în baza 8 ca fiind $27{,}2_8$.

Generalizare
Pentru a converti un număr scris în baza 2 într-o bază de numeraţie de forma 2^k se completează numărul cu cifre 0, astfel încât atât numărul de cifre al părţii întregi, cât şi numărul de cifre al părţii fracţionare să fie multiplu de *k*, apoi aceste grupuri de câte k cifre binare se înlocuiesc cu simbolul corespunzător din tabelul de codificare.

B. Algoritm de conversie din baza 2^k în baza 2
Pentru a converti un număr reprezentat în baza 2^k în reprezentarea corespunzătoare bazei 2, se efectuează o simplă înlocuire a simbolurilor bazei 2^k cu grupul de *k* cifre binare care corespunde simbolului conform tabelului prezentat. Atunci când în tabel nu sunt trecute *k* cifre binare, reprezentarea se va completa cu 0-uri în faţă până la obţinerea a exact *k* cifre. În final 0-urile nesemnificative vor fi ignorate.

Exemplu

$$4A5{,}8_{16} = \underbrace{0100}_{4}\ \underbrace{1010}_{A}\ \underbrace{0101}_{5},\underbrace{1000}_{8}{}_2 = 10010100101{,}1_2$$

Observaţie
Un număr dat în baza b_1 poate fi convertit direct în baza b_2 (ambele baze fiind diferite de baza 10) fără a se intermedia trecerea prin baza de numeraţie 10. Algoritmul este acelaşi ca în cazul conversiei în baza 10, cu deosebirea că toate calculele se vor efectua în baza b_2.

5.3. Operaţii cu numere naturale în diverse baze de numeraţie

5.3.1. Adunarea

Pentru a aduna două numere naturale scrise într-o bază de numeraţie *b*, vom aduna cifrele de acelaşi rang, de la dreapta la stânga. Dacă numărul rezultat din adunarea a două cifre se reprezintă cu o singură cifră, atunci am obţinut cifra rezultat. Dacă rezultatul adunării este format din două simboluri, atunci restul împărţirii la *b* este cifra rezultat şi vom avea cifră de transport (în cazul a două numere aceasta este egal cu 1).

Exemplu

2367_8+	Adunăm cifrele de rang 0: $7_8 + 7_8 = 7_{10} + 7_{10} = 14_{10} = 1 \cdot 8_{10} + 6_{10} = 1\mathbf{6}_8$
$\underline{3707_8}$	Cifra **6** este cifra de rang 0 în rezultat, **1** este cifră de transport.
6276_8	Adunăm cifrele de rang 1: $6_8 + 0_8 + 1_8 = 7_{10} = \mathbf{7}_8$
	Cifra **7** este cifra de rang 1 în rezultat, nu avem cifră de transport.
	Adunăm cifrele de rang 2: $3_8 + 7_8 = 10_{10} = 1 \cdot 8 + 2 = 1\mathbf{2}_8$
	Cifra **2** este cifra de rang 1 în rezultat, **1** este cifră de transport.
	Adunăm cifrele de rang 3: $2_8 + 3_8 + 1_8 = 6_{10} = \mathbf{6}_8$
	Cifra **6** este cifra de rang 3 în rezultat, nu avem cifră de transport.

5.3.2. Scăderea

Operația de scădere se efectuează asemănător scăderii în baza 10, cu deosebirea că unitatea împrumutată va conține *b* subunități şi nu 10.

Exemplu

6276_8–	Calculăm cifra de rang 0: ca să putem efectua 6 – 7, împrumutăm 1 din cifra de rang 1, obținem $8 + 6 - 7 = \mathbf{7}_8$
$\underline{2367_8}$	
3707_8	Calculăm cifra de rang 1: $(7 - 1) - 6 = \mathbf{0}$
	Calculăm cifra de rang 2: ca să putem efectua 2 – 3, împrumutăm 1 din cifra de rang 3, obținem $8 + 2 - 3 = 7_8$
	Calculăm cifra de rang 3: $(6 - 1) - 2 = 3_8$

5.3.3. Înmulțirea

Înmulțirea a două numere scrise într-o bază de numerație *b* se efectuează asemănător înmulțirii în baza 10, dar se va ține cont de tabla adunării şi înmulțirii în baza *b*.

Exemplu

Tabla înmulțirii în baza 4 se completează efectuând calculele în baza 4. De exemplu, $3_4 \cdot 3_4 = 3_{10} \cdot 3_{10} = 9_{10} = 21_4$.

	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

Vom calcula $103_4 \times 13_4$:

$\underline{103_4} \times 13_4$	Calculăm cifra de rang 0: $3_4 \cdot 3_4 = 9_{10} = 21_4$, scriem **1**, transport 2
321	Calculăm cifra de rang 1: $3_4 \cdot 0_4 + 2_4 = 2_4$, scriem **2**, transport 0
$\underline{103\ }$	Calculăm cifra de rang 2: $3_4 * 1_4 + 0 = 3_4$, scriem **3**, transport 0
2011_4	$103_4 \cdot 1 = \mathbf{103_4}$; adunând 321_4 cu 1030_4, obținem 2011_4.

5.3.4. Împărțirea

Pentru efectuarea împărțirii a două numere scrise în baza *b*, vom utiliza aceleaşi reguli din sistemul zecimal, ținând cont de tabla înmulțirii şi tabla adunării în baza *b*.

Exemplu

$2011_4 : 13_4 = 103_4$	13_4 se cuprinde în 20_4 de **1** ori.
$\underline{13}$	$20_4 - 13_4 = 1_4$
=11	se coboară următoarea cifră: 13 în 11 se cuprinde de **0** ori.
$\underline{00}$	se mai coboară o cifră: 13 în 111 se cuprinde de **3** ori
111	$3_4 \cdot 13_4 = 111_4$
$\underline{111}$	
===	

5.4. Criterii de divizibilitate în diverse baze de numerație

Considerăm ca fiind importantă cunoaşterea şi utilizarea criteriilor de divizibilitate în pregătirea elevilor pentru concursurile şcolare de programare, motiv pentru care am inclus această prezentare. Rezultatele au fost preluate (fără demonstrațiile aferente) din lucrarea **Sisteme de numerație**, autor *Ilie Diaconu*, Editura Studium, 1996. Cei care doresc să aprofundeze această temă pot consulta bibliografia de specialitate.

În cele ce urmează notăm cu *b* o bază de numerație oarecare.

5.4.1. Divizibilitatea cu numărul *b* – 1

Un număr natural scris în baza *b* se divide cu $b - 1$ dacă şi numai dacă suma cifrelor sale este un număr multiplu de $b - 1$.

5.4.2. Divizibilitatea cu *b* + 1

Un număr natural scris în baza *b* se divide cu $b + 1$ dacă şi numai dacă diferența dintre suma cifrelor de ordin impar şi suma cifrelor de ordin par este multiplu de $b + 1$.

5.4.3. Divizibilitatea cu *b* în baze de forma *b* · *k* + 1

Un număr natural scris în baza $b \cdot k + 1$ se divide cu *b* dacă şi numai dacă suma cifrelor sale este un număr multiplu de *b*.

5.4.4. Divizibilitatea cu *b* în baze de forma *b* · *k* – 1

Un număr natural scris în baza $b \cdot k - 1$ se divide cu *b* dacă şi numai dacă diferența dintre suma cifrelor de ordin impar şi suma cifrelor de ordin par este multiplu de *b*.

5.4.5. Divizibilitatea cu b, b^2, b^3,…, în baza b

Un număr natural scris în baza b se divide cu b, b^2, b^3,… dacă şi numai dacă numărul se termină cu 0, doi de 0, ….

5.4.6. Divizibilitatea cu b în baze de forma $b \cdot k$

Un număr natural scris în baza $b \cdot k$ se divide cu b dacă şi numai dacă ultima cifră a numărului este un multiplu de b.

5.4.7. Divizibilitatea cu $b \cdot k - 1$ sau $b \cdot k + 1$ în baza b

Un număr natural scris în sistemul de numeraţie având baza b se divide cu $b \cdot k - 1$ / $b \cdot k + 1$ dacă şi numai dacă suprimându-i ultima cifră şi scăzând, respectiv adunând de k ori cifra suprimată se obţine un număr divizibil cu $b \cdot k - 1$, respectiv $b \cdot k + 1$.

5.4.8. Criteriul de divizibilitate cu 2 în diverse baze de numeraţie

Un număr natural este divizibil cu 2 în baza de numeraţie:

a) $b = 2k$ dacă şi numai dacă ultima sa cifră este multiplu de 2;

b) $b = 2k + 1$ dacă şi numai dacă suma cifrelor sale este multiplu de 2.

5.4.9. Criteriul de divizibilitate cu 3 în diverse baze de numeraţie

Un număr natural este divizibil cu 3 în baza de numeraţie:

a) $b = 3k$ dacă şi numai dacă ultima sa cifră este multiplu de 3

b) $b = 3k + 1$ dacă şi numai dacă suma cifrelor sale este multiplu de 3;

c) $b = 3k - 1$ dacă şi numai dacă diferenţa dintre suma cifrelor de ordin impar şi suma cifrelor de ordin par este multiplu de 3

5.4.10. Criteriul de divizibilitate cu 4 în diverse baze de numeraţie

Un număr natural este divizibil cu 4 în baza de numeraţie:

a) $b = 4k$ dacă şi numai dacă ultima sa cifră este multiplu de 4;

b) $b = 4k + 1$ dacă şi numai dacă suma cifrelor sale este multiplu de 4;

c) $b = 4k + 2$ dacă şi numai dacă suma dintre dublul penultimei cifre şi ultima cifră este multiplu de 4;

d) $b = 4k - 1$ dacă şi numai dacă diferenţa dintre suma cifrelor de ordin impar şi suma cifrelor de ordin par este multiplu de 4; (cifrele se consideră numărate de la dreapta la stânga).

5.4.11. Criteriul de divizibilitate cu 5 în diverse baze de numerație

Numărul natural $n = a_n a_{n-1} \ldots a_1 a_0$, scris în baza b se divide cu 5 în sistemele de numerație cu baza:

1. $b = 5k$, $k \in N^*$, dacă şi numai dacă ultima cifră se divide cu 5;
2. $b = 5k + 1$, $k \in N^*$, dacă şi numai dacă suma tuturor cifrelor se divide cu 5;
3. $b = 5k + 2$, $k \in N^*$, dacă şi numai dacă $(a_0 - a_2 + a_4 - \ldots) + 2(a_1 - a_3 + a_5 - \ldots)$ se divide cu 5;
4. $b = 5k + 3$, $k \in N^*$, dacă şi numai dacă $(a_0 - a_2 + a_4 - \ldots) + 3(a_1 - a_3 + a_5 - \ldots)$ se divide cu 5;
5. $b = 5k + 4$, $k \in N^*$, dacă şi numai dacă $(a_0 - a_2 + a_4 - \ldots) - (a_1 + a_3 + a_5 + \ldots)$ se divide cu 5.

5.5. Implementări sugerate

Pentru a vă familiariza cu modul în care se rezolvă problemele în care intervin implementări ale conversiilor, vă sugerăm să încercați să realizați algoritmi pentru:

1. conversie din baza 10 în baza 2;
2. conversie din baza 2 în baza 10;
3. conversie din baza 2 în baza 8 (16);
4. conversie din baza 8 şi 16[*)] în baza 2;
5. conversie din baza 10 în baza $b < 10$;
6. conversie din baza $b < 10$ în baza 10;
7. conversie din baza $a < 10$ în baza $b < 10$;
8. conversie din baza 16 în baza 10[*)];
9. conversie din baza 10 în baza 16[*)];
10. conversie din baza $a < 36$ în baza $b < 36$[*)];
11. conversie din baza 10 în baza b (oricât de mare)[*)];
12. conversie din baza b (oricât de mare) în baza 10[*)];
13. conversie din baza a (oricât de mare) în baza b (oricât de mare)[*)].

[*)] Se pot amâna după învățarea prelucrării şirurilor de caractere.

5.6. Probleme propuse

5.6.1. Conversie

Să se convertească un număr natural n, scris într-o bază de numerație b_1, într-o altă bază de numerație b_2. Înainte de a se efectua conversia, se va verifica dacă numărul dat este corect scris (în scrierea lui s-au folosit cifre permise de baza de numerație în care se presupune că a fost scris).

Date de intrare

Cele două baze de numerație b_1 și b_2 se citesc de pe prima linie a fișierului de intrare **`CONV.IN`**. Pe linia următoare se află numărul natural care trebuie convertit.

Date de ieșire

Numărul convertit se va scrie în fișierul **`CONV.OUT`**. În cazul în care numărul nu este corect scris, în fișierul de ieșire se va scrie mesajul `'numar incorect'`.

Restricții și precizări

- numerele vor avea cel mult 9 cifre în ambele baze b_1 și b_2
- $2 \leq b_1, b_2 \leq 10$.

Exemple

CONV.IN	**CONV.OUT**
`2 8` `10101010`	`252`
CONV.IN	**CONV.OUT**
`5 9` `4001252`	`numar incorect`

5.6.2. Operații într-o bază de numerație dată

Să se efectueze operația de adunare sau de scădere (conform cerinței din fișierul de intrare) a două numere naturale date, reprezentate într-o bază de numerație dată b, fără să se efectueze conversii în baza 10.

Date de intrare

Pe prima linie a fișierului de intrare **`OPER.IN`** se află un număr natural b, reprezentând baza de numerație în care se va efectua operația. Pe următoarele linii ale aceluiași fișier se află primul număr, apoi operatorul + sau – , iar pe ultima linie se află cel de-al doilea număr.

Date de ieşire
Rezultatul operaţiei se va scrie în fişierul **`OPER.OUT`**.

Restricţii şi precizări
- $1 \leq număr_1, număr_2 \leq 100000000_b$;
- $număr_1 \geq număr_2$;
- $2 \leq b \leq 10$.

Exemplu

`OPER.IN`	**`OPER.OUT`**
3 12 + 11	100

5.6.3. Criterii de divizibilitate într-o bază de numeraţie dată

Se consideră o bază de numeraţie *b* şi un număr natural scris în această bază. Să se verifice dacă numărul dat este divizibil cu 2, 3, 4 şi 5, utilizând criterii de divizibilitate.

Date de intrare
Pe prima linie a fişierului de intrare **`NR.IN`** se află un număr natural *b*, reprezentând baza de numeraţie. Pe linia următoare se află numărul care urmează să fie analizat.

Date de ieşire
În fişierul **`NR.OUT`** se vor scrie cifrele 2, 3, 4 şi/sau 5 în cazul în care numărul dat este divizibil cu acestea. În situaţia în care numărul este divizibil cu două sau mai multe numere dintre cele enumerate, acestea vor fi separate prin câte un spaţiu. Dacă numărul nu este divizibil cu nici unul dintre numerele 2, 3, 4 şi 5, în fişierul de ieşire se va scrie `'NU'`.

Restricţii şi precizări
- $1 \leq n \leq 1000000000_b$;
- $2 \leq b \leq 10$.

Exemple

`NR.IN`	**`NR.OUT`**
3 120	3 5
`NR.IN`	**`NR.OUT`**
10 311	NU

5.6.4. Sistemul de numerație *Fibonacci*

Cifrele sistemului de numerație *Fibonacci* sunt 0 şi 1. În acest sistem de numerație se pot reprezenta numerele asemănător sistemului zecimal. Numărul $m = c_n c_{n-1} \ldots c_3 c_2 c_1$, scris în baza *Fibonacci*, în sistemul zecimal are valoarea $c_n \cdot a_n + \ldots + c_2 \cdot 2 + c_1 \cdot 1$, unde a_i sunt numere *Fibonacci*.

Să se scrie un program care converteşte un număr dat în baza 10 în baza *Fibonacci* şi un număr dat în baza *Fibonacci* în sistemul zecimal.

Date de intrare

Pe prima linie a fişierului de intrare **`FIBO.IN`** se află un număr natural *m*, reprezentat în baza *Fibonacci*. Pe linia următoare se află un număr în baza 10.

Date de ieşire

Pe prima linie a fişierului **`FIBO.OUT`** se va scrie reprezentarea primului număr din fişierul de intrare, reprezentat în baza 10. Pe cea de a doua linie se va scrie reprezentarea celui de-al doilea număr în baza *Fibonacci*.

Restricții şi precizări

- $1 \leq nr_1, nr_2 \leq 1000000000$.

Exemple

FIBO.IN	**NR.OUT**
10101001	53
53	10101001

5.7. Soluțiile problemelor propuse

Înainte de a prezenta rezolvările problemelor, dorim să clarificăm următoarele: în calculator orice număr este reprezentat în baza 2, iar „imaginea” pe care o vedem, afişând un astfel de număr este ***un număr reprezentat în baza 10***. În concluzie, atunci când în aceste rezolvări vorbim despre conversii, de fapt construim „imaginea” numerelor formate din cifre conform sistemului de numerație. Din acest motiv, în continuare prin conversii înțelegem construirea numerelor în baza 10 care „arată” *ca şi cum ar fi în baza b*.

5.7.1. Conversie

Pentru a converti un număr dintr-o bază de numerație în alta există, în principiu, două posibilități:

- se converteşte numărul în baza 10 şi apoi acesta se converteşte în baza dorită;
- se converteşte numărul în baza de numerație dorită prin aplicarea algoritmului de conversie clasic, cu condiția că toate calculele se vor efectua în baza specificată.

Algoritmul descris în pseudocod foloseşte prima metodă. Dacă baza b_1 este baza 10, nu este necesară conversia numărului din baza b_1 în baza 10 şi similar, dacă b_2 este egal cu 10, nu este nevoie să convertim rezultatul parţial.

```
Algoritm Conversii:
  citeşte b1,b2                    { se citesc cele două baze de numeraţie }
  citeşte număr                                   { se citeşte numărul dat }
                          { se verifică dacă numărul este scris corect în baza dată }
  valid ← adevărat                 { presupunem că numărul este corect scris }
  nr ← număr                                     { copie a numărului dat }
  cât timp valid şi (nr ≠ 0) execută:
    dacă rest[nr/10] ≥ b1 atunci
      valid ← fals      { numărul corect scris are doar cifre mai mici decât b1 }
    sfârşit dacă
    nr ← [nr/10]
  sfârşit cât timp
  dacă nu valid atunci
    scrie 'numar incorect')
  altfel
                                          { conversie din baza b1 în baza 10 }
    dacă b1 = 10 atunci
      parţial ← număr
    altfel
                  { dacă b1 = 10, nu este necesară etapa de conversie în baza 10 }
      parţial ← 0              { rezultatul parţial al conversiei (din b1 în 10) }
      p ← 1                                     { p păstrează puterile bazei }
      cât timp număr ≠ 0 execută:
        cifra ← rest[număr/10]
        număr ← [număr/10]
        parţial ← parţial + cifra*p
        p ← p*b1
      sfârşit cât timp
    sfârşit dacă
    dacă b2 = 10 atunci            { dacă b2 = 10, nu este necesară conversia }
      rezultat ← parţial
    altfel                                { conversie din baza 10 în baza b2 }
      rezultat ← 0
      p ← 1                                  { p păstrează puterile bazei 10 }
      repetă
        cifra ← rest[parţial/b2]
        parţial ← rest[parţial/b2]
```

```
        rezultat ← rezultat + cifra*p
        p ← p*10
      până când parţial = 0
    sfârşit dacă
    scrie rezultat
  sfârşit dacă
sfârşit algoritm
```

5.7.2. Operaţii într-o bază de numeraţie dată

În funcţie de operaţia specificată în fişier (adunare sau scădere), algoritmul va trebui să efectueze suma sau diferenţa numerelor date, operaţiile efectuându-se în baza de numeraţie în care au fost scrise numerele.

Se vor urmări aceste operaţii pe un exemplu:

Fie b = 3, *număr*$_1$ = 21022_3 şi *număr*$_2$ = 2011_3.

Dacă *operator* = `'+'`

operaţia	transport vechi	calcule	cifră rezultată	transport
21022_3+	0	$0+2_3+1_3=3_{10}=10_3$	0	1
$\underline{2011_3}$	1	$1+2_3+1_3=4_{10}=11_3$	1	1
100110_3	1	$1+0_3+0_3=1_{10}=1_3$	1	0
	0	$0+1_3+2_3=3_{10}=10_3$	0	1
	1	$1+2_3+0_3=3_{10}=10_3$	0	1
	1	$1+0_3+0_3=1_{10}=1_3$	1	0

Dacă operator = `'-'`

operaţia	împrumut	calcule	cifra rezultată	transport
21022_3-	0_{10}	$0_{10}+2_3-1_3=1_3$	1_3	0
$\underline{2011_3}$	0_{10}	$0_{10}+2_3-1_3=1_3$	1_3	0
12011_3	0_{10}	$0_{10}+0_3-0_3=0_3$	0_3	0
	3_{10}	$3_{10}+1_3-2_3=2_3$	2_3	0
	0_{10}	$0_{10}+(2_3-1_3)=1_3$	1_3	-1

Algoritmul care descrie aceste operaţii este prezentat în continuare.

```
Algoritm Operaţii:
  citeşte b                { baza de numeraţie în care se efectuează operaţia }
  citeşte număr1
  citeşte operator                                          { '+' sau '-' }
  citeşte număr2
  rezultat ← 0                                  { iniţializarea rezultatului }
```

```
  transport ← 0                          { inițializarea cifrei de transport }
  p ← 1                                  { p se inițializează cu 10 la puterea 0 }
  cât timp (număr1 ≠ 0) sau (număr2 ≠ 0) execută:
                                  { cât mai există cifre cel puțin într-un număr }
    dacă operator = '+' atunci
                                  { se adună ultimele cifre ale celor două numere }
      parțial ← transport + rest[număr1/10] + rest[număr2/10]
      dacă parțial ≥ b atunci
        parțial ← parțial - b
        transport ← 1
      altfel transport ← 0
      sfârşit dacă
    altfel                        { se scad ultimele cifre ale celor două numere }
      parțial ← rest[număr1/10] - transport – rest[număr2/10]
      dacă parțial < 0 atunci                     { este nevoie de împrumut }
        parțial ← parțial + b
        transport ← 1
      altfel transport ← 0
      sfârşit dacă
    sfârşit dacă
    rezultat ← rezultat + p*parțial
                                      { cifra obținută se adaugă la rezultat }
    p ← p*10                     { se pregăteşte puterea pentru pasul următor }
    număr1 ← [număr1/10]                       { se trece la următoarea cifră }
    număr2 ← [număr2/10]
  sfârşit cât timp
  dacă transport > 0 atunci
    rezultat ← rezultat + p                    { ultima cifră de transport }
  sfârşit dacă
  scrie rezultat
sfârşit algoritm
```

5.7.3. Criterii de divizibilitate într-o bază de numerație dată

Programul poate fi realizat verificând criteriile de divizibilitate cu 2, 3, 4 şi 5, criterii care au fost prezentate în suportul teoretic al lecției. Condițiile au fost grupate după forma bazei de numerație, deoarece se observă similitudini în definirea acestor criterii. De exemplu, atunci când baza de numerație este multiplu de *cifra*, toate criteriile cer condiția ca ultima cifră a numărului să fie multiplu de *cifra* (s-a notat cu *cifra* valoarea 2, 3, 4 sau 5). De asemenea, în cazul în care baza de numerație este de forma $k \cdot cifra$, toate criteriile pun condiția ca suma cifrelor numărului să se dividă la *cifra*.

Se descrie în continuare un algoritm de rezolvare diferit de cel prezentat mai sus. În subalgoritm vom trata doar divizibilitatea cu 5 a unui număr reprezentat în baza *b*, deoarece criteriul prezentat în partea teoretică este valabil doar pentru baze mai mari sau egale cu 5. Nu vom genera număr în baza 10, doar ultima cifră ne interesează, dar calculele trebuie să le efectuăm în baza 10, deoarece *baza* este mai mică decât 5.

```
Subalgoritm verif(număr):
  nn ← număr
  putere ← 1
  cât timp nn > 9 execută:        { traversăm cifrele numărului şi pregătim }
    nn ← [nn/10]          { putere pentru a obţine puterea cu care vom împărţi }
    putere ← putere*10
  sfârşit cât timp
  cifră ← [număr/putere]
  cât timp putere > 10 execută: { traversăm cifrele de la stânga la dreapta }
    putere ← [putere/10]
    cifră ← rest[(cifră*baza + [număr/putere])/10]
  sfârşit cât timp
  dacă (cifră = 0) sau (cifră = 5) atunci
    scrie 'Numarul se divide cu 5'
  sfârşit dacă
sfârşit subalgoritm
```

5.7.4. Sistemul de numeraţie *Fibonacci*

Algoritmul de conversie se bazează pe descompunerea unui număr natural în sumă de numere *Fibonacci*, prezentat în capitolul 4.

Subalgoritmul următor transformă un număr dat în sistemul de numeraţie *Fibonacci* (*nn*) în număr zecimal (*n*). Vom căuta în numărul *nn* cifre egale cu 1, deoarece ele reprezintă prezenţa în suma care reprezintă numărul *n*, a unui număr Fibonacci. În număr doar anumiţi termeni din şir participă, dar trebuie să-i generăm pe toţi, până când împărţirea cu 10 se finalizează cu un cât egal cu 0.

```
Subalgoritm DinFiboÎn10(nn,n):
          { nn: număr în sistemul de numeraţie Fibonacci, n: număr în baza 10 }
  n ← 1
  dacă nn ≠ 1 atunci
    dacă rest[nn/10] = 0 atunci
      n ← 0
    sfârşit dacă
    nn ← [nn/10]
```

```
    a ← 1
    b ← 1
    c ← a + b
    cât timp nn > 0 execută:                        { cât timp mai avem cifre }
      cifra ← rest[nn/10]
      dacă cifra = 1 atunci
                  { cifra 1 aduce un număr Fibonacci în valoarea numărului n }
        n ← n + c
      sfârşit dacă
        a ← b
        b ← c
        c ← a + b
        nn ← [nn/10]
    sfârşit cât timp
  sfârşit dacă
sfârşit subalgoritm
```

Subalgoritmul următor transformă un număr zecimal în număr reprezentat în sistemul de numerație *Fibonacci*. Dacă numărul dat este număr *Fibonacci*, înseamnă că acesta (t_i) s-a obținut adunând doi termeni consecutivi (t_{i-1} şi t_{i-2}) din şir. În acest caz t_i nu are o reprezentare unică, deoarece poate fi scris şi sub forma m = 100...0, unde numărul 0-urilor este egal cu numărul numerelor Fibonacci mai mici decât numărul dat, dar şi sub forma 1100..0 unde numărul 0-urilor este egal cu numărul numerelor Fibonacci mai mici decât t_{i-2}. Subalgoritmul va determina primul mod de afişare.

```
Subalgoritm Din10ÎnFibonacci(n,nn):
  nn ← 1                              { dacă n este egal cu 1, şi nn va fi 1 }
  dacă n ≠ 1 atunci
    a ← 1                           { începem generarea numerelor Fibonacci }
    b ← 1
    c ← a + b
    putere ← 1
    cât timp b < n execută:
      a ← b
      b ← c
      c ← a + b
      putere ← putere*10
    sfârşit cât timp
                    { dacă n este număr Fibonacci, nn este de forma 1 · putere }
    nn ← putere
```

```
    dacă c ≠ n atunci
  { dacă n nu este număr Fibonacci, îl descompunem în sumă de numere Fibonacci }
      n ← n - b
      cât timp n > 0 execută:                 { cât timp mai avem rest }
        cât timp b > n execută:
          c ← b
          b ← a
          a ← c - b
          putere ← [putere/10]
        sfârşit cât timp
        nn ← nn + putere
        n ← n - b
      sfârşit cât timp
    sfârşit dacă
  sfârşit dacă
sfârşit algoritm
```