Universitatea Babeș-Bolyai, Cluj-Napoca
Facultatea de Matematică și Informatică
Specializarea Informatică, română

Enăchioiu Teofana
Grupa 213

Documentație

Cuprins

Informaţii

Aplicaţia trebuie să exemplifice cele trei metode de conversie ale numerelor naturale (împărţiri succesive, substituţie şi utilizarea unei baze intermediare, de obicei, baza 10) între două baze de numeraţie diferite de 10, conversiile rapide între bazele puteri ale lui 2 (2, 4, 8, 16) şi operaţiile aritmetice într-o bază oarecare p (adunare, scădere, înmulţire cu o cifră şi împărţire la o cifră), fără a trece numărul prin baza 10 (p:{2,3,...,9,10,16}) —— *Deși această frază exprimă cerința, nu este enunțul problemei. Enunțul problemei ar trebui să specifice mai clar ce și cum va introduce utilizatorul și ce va obține.*

De exemplu: se dau două numere în baze diferite şi o a treia bază în care ele se vor aduna.

Sugestii:

- se va trece din baza mai mică în cea mare prin substituţie, iar din cea mare în cea mică prin împărţiri succesive, pentru a se utiliza doar împărţiri/înmulțiri cu o cifră.
- numerele se recomandă a se păstra în memorie prin şirul cifrelor.

Forma executabilă şi codul aplicaţiei vor fi inscripţionate pe CD-ul grupei, în care fiecare student va avea un director propriu. Documentaţiile vor fi predate pe hârtie cadrului didactic îndrumător de la seminar, împreună cu CD-ul grupei înainte de data de 1 decembrie. După această dată nu se mai primesc teme electronice. Documentaţiile vor respecta structura documentaţiilor de la Fundamentele Programării şi trebuie să conţină cel puţin: enunţul exact al aplicaţiei implementate, pseudocodul algoritmilor utilizaţi, consideraţii de implementare şi date de test.

Notarea se va face după următorul barem:

10% notă: punctul din oficiu

70% notă: aplicaţia (numele autorului se va găsi atât în cod, cât şi la execuţie)

1p existenţa algoritmului (în formă executabilă) de conversie prin împărţiri succesive

1p existenţa algoritmului (în formă executabilă) de conversie prin substituţie

1p existenţa algoritmului (în formă executabilă) de conversie utilizând o bază intermediară

(punctajele se înjumătăţesc dacă conversiile de mai sus au ca bază de pornire respectiv destinaţie obligatoriu baza 10, nefuncţionând direct dintr-o bază diferită de 10 într-o altă bază diferită de 10; şi se pierde un sfert din punctaj dacă nu se pot converti numere în/din baza 16)

2p existenţa algoritmilor (în formă executabilă) de conversii rapide din baza 2 în baza 4, 8 sau 16 şi respectiv invers

1p adunarea a două numere într-o bază oarecare

1p scăderea a două numere într-o bază oarecare

1p înmulţirea cu o cifră într-o bază oarecare

1p împărţirea la o cifră într-o bază oarecare

1p claritatea codului (identare, comentarii, nume de variabile sugestive)

Observaţii:

- dacă lipseşte sau nu funcţionează varianta executabilă a programului, atunci nota pe aplicaţie este 2

- dacă nu sunt mesaje clare cu privire la ce şi cum trebuie introdus de utilizator respectiv o prezentare clară a rezultatelor, atunci nota maximă pe aplicaţie este 4.

20% notă: documentaţia (numele autorului va fi scris clar, documentaţia se va scrie de mână, excepţie făcând doar cazurile speciale)

1p enunţul problemei

1p diagrama de apel a subalgoritmilor

1p specificarea tipurilor de date folosite

3p subalgoritmii principali vor fi specificaţi şi scrişi în pseudocod (date, rezultate, precondiţii, postcondiţii – 1p; pseudocodul – 2p)

3p cel puţin un set de date de test pentru întreaga aplicaţie eventual mai multe seturi diferite pentru părţile care necesită acest lucru

1p claritatea documentaţiei (structurată, scrisă frumos, ...)

Enunțul problemei

Să se realizeze o aplicație care permite următoarele operații:

- Adunarea a două numere într-o bază oarecare
- Scăderea a două numere într-o bază oarecare
- Înmulțirea cu o cifră într-o bază oarecare
- Împărțirea cu o cifră într-o bază oarecare
- Conversia unui număr dintr-o bază în alta (se utilizează conversia prin împărțiri, prin substituție sau conversiile rapide)
- Conversia unui număr utilizând o bază intermediară

Datele de intrare(bazele și numerele) vor fi citite de la tastatură și vor fi validate. Se operează cu numere naturale.

Diagrama de apel a subprogramelor

Apelul operației de adunare

Utilizator	Aplicație	Descriere
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	Se afișează meniul și se cere introducerea unei comenzi de la tastatura (1,2,3,4,5,x).
1	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a primului număr
16	-	Baza dorită
-	Numar=	Se cere introducerea primului număr
1F49C	-	Numărul dorit
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a celui de-al doilea număr
9	-	Baza dorită
	Numar=	Se cere introducerea celui de-al doilea număr
1884	-	Numărul dorit
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care se va efectua operația dorită
4	-	Baza dorită
-	1F49C(16) + 1884(9) = 133102130(4) + 112231(4) = 133221021(4)	Rezultatul efectuării operației

Apelul operației de scădere

Utilizator	Aplicație	Descriere
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza	Se afișează meniul și se cere introducerea unei comenzi de la tastatura (1,2,3,4,5,x).

	intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	
2	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a primului număr
16	-	Baza dorită
-	Numar=	Se cere introducerea primului număr
1F49C	-	Numărul dorit
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a celui de-al doilea număr
9	-	Baza dorită
	Numar=	Se cere introducerea celui de-al doilea număr
1884	-	Numărul dorit
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care se va efectua operația dorită
4	-	Baza dorită
-	1F49C(16) - 1884(9) = 133102130(4) - 112231(4) = 132323233(4)	Rezultatul efectuării operației

Apelul operației de înmulțire

Utilizator	Aplicație	Descriere
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	Se afișează meniul și se cere introducerea unei comenzi de la tastatura (1,2,3,4,5,x).
3	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a numărului
16	-	Baza dorită
-	Numar=	Se cere introducerea numărului
1F49C	-	Numărul dorit
-	Baza:	Se cere introducerea unui număr

		pentru baza de numerație a cifrei
15	-	Baza dorită
	Cifra=	Se cere introducerea cifrei
C	-	Numărul dorit
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care se va efectua operația dorită
14	-	Baza dorită
-	1F49C(16) * C(15) = 2C0640(14)	Rezultatul efectuării operației

Apelul operației de împărțire

Utilizator	Aplicație	Descriere
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	Se afișează meniul și se cere introducerea unei comenzi de la tastatura (1,2,3,4,5,x).
4	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a numărului
16	-	Baza dorită
-	Numar=	Se cere introducerea numărului
1F49C	-	Numărul dorit
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a cifrei
15	-	Baza dorită
	Cifra=	Se cere introducerea cifrei
C	-	Numărul dorit
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care se va efectua operația dorită
14	-	Baza dorită
-	1F49C(16) : C(15) = 3C6B(14) rest 8(14)	Rezultatul efectuării operației

Apelul operației de conversie

Utilizator	Aplicație	Descriere
-	Operatiile disponibile:	Se afișează meniul și se cere

	1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	introducerea unei comenzi de la tastatura (1,2,3,4,5,x).
5	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a numărului
16	-	Baza dorită
-	Numar=	Se cere introducerea numărului
1F49C	-	Numărul dorit
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care se va efectua operația dorită
7	-	Baza dorită
-	1F49C(16) = 1042430(7)	Rezultatul efectuării operației

Apelul operației de conversie utilizând bază intermediară

Utilizator	Aplicație	Descriere
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:	Se afișează meniul și se cere introducerea unei comenzi de la tastatura (1,2,3,4,5,x).
6	-	Comanda dorită
-	Baza:	Se cere introducerea unui număr pentru baza de numerație a numărului
16	-	Baza dorită
-	Numar=	Se cere introducerea numărului
1F49C	-	Numărul dorit
-	Baza:	Se cere introducerea unui număr pentru baza destinație a numărului
7	-	Baza dorită
-	Baza in care se va efectua operatia:	Se cere introducerea unui număr pentru baza de numerație în care

		se va efectua operația dorită
5	-	Baza dorită
-	1F49C(16) = 13100111(5) = 1042430(7)	Rezultatul efectuării operației

Tipuri de date folosite

Pentru bazele de numerație se utilizează numere naturale, mai mari decât 2.

În ceea ce privește numerele, ele se citesc de la tastatură ca string-uri, iar mai apoi se creează, pe baza lor, un nou tip de dată – **Număr.** Tipul *Număr* este format dintr-o listă ce conține numere, corespunzătoare cifrelor. (ex: A=10, F=15, 9=9). Operațiile se vor efectua asupra listei de cifre.

În final, lista de cifre, precum și rezultatul operației, vor constitui alt tip de date definit în program – **Rezultat.** Tipul *Rezultat* reface numărul, transformând numerele corespunzătoare cifrelor în caractere.

Algoritmii principali scriși în pseudocod

Algoritmul de adunare

Se determina lungimea maxima a listei cu cifrele sumei, apoi se inițializează cu 0 (neseminficative!). Se completează cele două numere de adunat cu zerouri pana la lungimea sumei. Se adună fiecare cifră de pe aceeași poziție, de la dreapta spre stânga, ținând cont de eventualul transport. La final, se elimină zerourile de la începutul listei de cifre a sumei.

Date de intrare: numar1, numar2 (liste de numere), baza (număr natural)
Date de ieșire: suma (listă de numere)

```
index = max(len(numar1),len(numar2)) // Lungimea maxima a sumei este data de index
diferenta = index-len(numar1)+1 // Numarul de zerouri ce trebuie adaugat
nr1=zerorizare(numar1,diferenta) // Functia "zerorizare" adauga zerouri nesemnificative la inceputul listei
diferenta = index-len(numar2)+1
nr2=zerorizare(numar2, diferenta)
t=0 // Transportul intial este 0
suma=[0]*(index+1) // Initializam suma cu 0
cat timp index>=0
    suma[index]=(nr1[index]+nr2[index]+t)%baza
    t=(nr1[index]+nr2[index]+t)//baza
    index=index-1
sfarsit_cat_timp
suma=eliminareZeroNesemnificativ(suma) // Se elimina zerourile nesemnificative
```

Algoritmul de scădere

Lungimea maxima a listei cu cifrele diferenței este egală cu lungimea listei primului număr. Se inițializează lista corespunzătoare rezultatului cu 0, iar cel de-al doilea număr se completează cu zerouri nesemnificative la început. Se scade din primul număr cel de-al doilea, ținând cont de eventualul împrumut. La final, se elimină zerourile de la începutul listei de cifre a diferenței.

Date de intrare: numar1, numar2 (liste de numere), baza (număr natural)
Date de ieșire: diferența (listă de numere)
Precondiții: numar1>=numar2
Postcondiții: diferența >=0

```
index=len(numar1)-1 // Lungimea maxima a listei de cifre a diferentei
nr1=numar1 // Descazutul
nr2=zerorizare(numar2, index-len(numar2)+1) // Scazatorul se zerorizeaza la inceput cu zerouri neseminficative
t=0 // Transportul initial e 0
diferenta=[0]*(index+1) // Initializam lista cu 0
```

```
cat timp index>=0
    daca nr1[index]+t >= nr2[index]
         atunci//Nu e nevoie de imprumut
              diferenta[index]=nr1[index]+t-nr2[index]
              t=0
         altfel//Trebuie imprumut
              diferenta[index]=baza+nr1[index]+t-nr2[index]
              t=-1
    sfarsit_daca
    index=index-1
sfarsit_cat_timp
diferenta=eliminareZeroNesemnificativ(diferenta) // Se elimina zerourile
nesemnificative
```

Algoritmul de înmulțire

Se inițializează o lista de cifre pentru rezultatul înmulțirii. Se înmulțește numărul la cifră, ținând cont de eventualul transport. La final, se elimină zerourile de la începutul listei de cifre a produsului.

Date de intrare: numar1 (liste de numere), numar2 (număr natural), baza (număr natural)
Date de ieșire: produs (listă de numere)
Precondiții: numar2 <baza, numar2!=0

```
t=0 // Transportul initial e 0
produs=[] // Initilizam o lista unde vor memora cifrele rezultatului
cat timp len(numar1)>0
    cifra=numar1[-1] // Se ia cifra de pe ultima pozitie
    term=(cifra*numar2+t)%baza
    t=(cifra*numar2+t)//baza
    produs=produs+term // Se adauga cifra la rezultatul final
    numar1=numar1[:-1] // Se elimina ultimul numar si se porneste in
continuare cu noul numar format
sfarsit_cat_timp
produs=produs+[t] // Se adauga transportul
produs.reverse() // Se inversează lista de numere
produs=eliminareZeroNesemnificativ(produs) // Se elimină zerourile
nesemnificative
```

Algoritmul de împărțire

Se inițializează o lista de cifre pentru rezultatul împărțirii. Se împarte număr1 la numar2 (care e o cifra, de fapt), ținând cont de eventualul transport. La final, se elimină zerourile de la începutul listei de cifre a catului.

Date de intrare: numar1 (liste de numere), numar2 (număr natural), baza (număr natural)
Date de ieșire: cat,rest (liste de numere)

Precondiții: numar2 <baza, numar2!=0

Postcondiții: cat >=0, rest<numar2

```
t=0
listaCat=[] // Initializam o lista de cifre pentru rezultatul final
cat timp len(numar1)>0
    cifra=numar1[0] // Luam prima cifra din numar
    cat=(t*baza+cifra)//numar2
    listaCat=listaCat+[cat] // Adaugam catul la rezultatul final
    rest=(t*baza+cifra)%numar2
    t=rest // Transportul ia valorea restului
    numar1=numar1[1:] // Se elimina prima cifra si se porneste de la noul
numar creat
sfarsit_cat_timp
listaCat=eliminareZeroNesemnificativ(listaCat) // Se elimina zerourile
nesemnificative
rest=[rest] // Valoarea restului
```

Algoritmul de conversie prin împărțiri succesive

Se inițializează o lista de cifre pentru rezultatul final al conversiei. Se împarte catul la baza destinație, până când acesta devine 0. Operația se realizează în baza sursă. Resturile luat în ordine inversă formează numărul.

Date de intrare: numar (listă de numere), bazaS, bazaD (numere naturale)

Date de ieșire: numarNou(listă de numere)

Precondiții: bazaS>bazaD

```
numarNou=[] // Initializam o lista de cifre pentru rezultatul final
cat=numar
cat_timp cat!=[0] // Deimpartinul diferit de 0
    rez=impartire(cat,bazaD,bazaS) // Se imparte catul la baza destinatie,
tinand cont de faptul ca baza in care se efectueaza operatia este baza sursa
    cat=rez['cat']
    eliminareZeroNesemnificativ(cat) // Se elimina zerourile nesemnificative
    numarNou=numarNou+rez['rest'] // Se adauga restul impartirii la
rezultatul final
sfrasit_cat_timp
numarNou.reverse() // Se inverseaza ordinea elementelor in lista (resturile)
```

Algoritmul de conversie prin substituție

Se inițializează o lista de cifre pentru rezultatul final al conversiei. Se înmulțește cifra de pe poziția *i* cu bazaS la puterea *i*. Operația se realizează în baza destinație.

Date de intrare: numar (listă de numere), bazaS, bazaD (numere naturale)

Date de ieșire: numarNou(listă de numere)

Precondiții: bazaS<bazaD

```
numarNou=[] // Se initializeaza lista de cifre corespunzatoare rezultatului final
p=[1] // Reprezinta bazaS la puterea 0
pentru i=len(numar)-1,0
     termen=inmultire(p, numar[i], bazaD) // Se inmulteste puterea lui bazaS cu cifra de pe pozitia corespunzatoare
     p=inmultire(p, bazaS, bazaD) // Se inmulteste puterea lui bazaS cu bazaS
     numarNou=adunare(numarNou,termen,bazaD) //Se aduna termenul calculat la rezultatul final
sfarsit_pentru
```

Algoritmul de conversie rapidă din baza 2 în baza 4

Se definește un dicționar ce conține grupuri de cifre ce reprezintă echivalențele între bazele 2 și 4. Se iau grupuri de câte doua cifre din număr și se convertesc pe baza dicționarului. Daca lungimea liste de cifre nu e divizibilă cu 2, se adaugă zerouri nesemnificative.

Date de intrare: numar (listă de numere)

Date de ieșire: numarNou(listă de numere)

Precondiții: bazaSursa=2, bazaDestinatie=4

```
conversieRapida={'[0, 0]':[0],'[0, 1]':[1],'[1, 0]':[2],'[1, 1]':[3]} //Se creaza un dictionar cu cheile egale cu grupurile de convertit.
numarNou=[] // Se initializeaza o noua lista
diferenta=len(numar)%2
daca diferenta>0 // Se verifica daca trebuie adaugate zerouri nesemnificative
    atunci
         numar=zerorizare(numar,2-diferenta) // Se adauga zerouri
sfarsit_daca
cat timp len(numar)>0
    cifre=numar[:2] // Se iau primele doua cifre din numar
    numarNou=numarNou+conversieRapida[str(cifre)] // Se convertesc cifrele pe baza dictionarului
    numar=numar[2:] // Se elimina cele doua cifre
sfarsit_cat_timp
```

Algoritmul de conversie rapidă din baza 2 în baza 8, respectiv 16 se construiește analog cu cel de sus.

Algoritmul de conversie rapidă din baza 16 în baza 2

Se definește un dicționar ce conține grupuri de cifre ce reprezintă echivalențele între bazele 16 și 2. Se ia fiecare cifră a numărului din baza 16 și se convertește pe baza dicționarului.

Date de intrare: numar (listă de numere)

Date de ieșire: numarNou(listă de numere)

Precondiții: bazaSursa=16, bazaDestinatie=2

```
conversieRapida={'0':[0, 0, 0, 0],'1':[0, 0, 0, 1],'2':[0, 0, 1, 0],'3':[0,
0, 1, 1],'4':[0, 1, 0, 0],'5':[0, 1, 0, 1],'6':[0, 1, 1, 0],'7':[0, 1, 1,
1],'8':[1, 0, 0, 0],'9':[1, 0, 0, 1],'10':[1, 0, 1, 0],'11':[1, 0, 1,
1],'12':[1, 1, 0, 0],'13':[1, 1, 0, 1],'14':[1, 1, 1, 0],'15':[1, 1, 1, 1]}
// Dictionarul cu echivalentele dintre baze
numarNou=[] // Se initializeaza o noua lista
cat timp len(numar)>0
    cifre=numar[0] // Se ia prima cifra din numar
    numarNou=numarNou+conversieRapida[str(cifre)] // Se converteste cifra pe
baza dictionarului
    numar=numar[1:] // Se elimina prima cifra din numar si se porneste cu
noul numar format
sfarsit_cat_timp
    numarNou=eliminareZeroNesemnificativ(numarNou) // Se elimina zerourile
nesemnificative
```

Algoritmul de conversie rapidă în baza 2 din baza 4, respectiv 8 se construiește analog cu cel de sus.

Algoritmul de conversie utilizând bază intermediară

Se convertește numărul din baza sursă în baza intermediară, utilizând conversiile rapide, substituția sau împărțirile succesive, apoi se convertește din baza intermediară în baza destinație, folosind aceleași metode.

Date de intrare: numar (listă de numere), bazaS, bazaD, bazaOp(numere naturale)

Date de ieșire: numarNou(listă de numere)

```
numar_convert1=conversie(numar,bazaS,bazaOp) // Conversia din baza sursa in
baza intermediara
numar_convert2=conversie(numar,bazaOp,bazaD) // Conversia din baza
intermediara in baza destinatie

//Functia conversie apeleaza corespunzator tipul de conversie (in functie de
baza sursa, baza intermediara si baza destinatie)
```

Date de intrare: numar (lista de numere), bazaS, bazaD (numere naturale)

Date de ieșire: numarul convertit

```
daca bazaS==bazaD // Nu se face conversie
      return numar
sfarsit_daca
daca bazaS==2 // Conversii rapide din baza 2
```

```
      atunci
            daca bazaD==4
                  atunci return conversie2to4(numar)
            sfarsit_daca
            daca bazaD==8
                        atunci return conversie2to8(numar)
            sfarsit_daca
            daca bazaD==16
                        atunci return conversie2to16(numar)
            sfarsit_daca
sfarsit_daca
daca bazaD==2 // Conversii rapide in baza 2
      atunci
            daca bazaS==4
                        atunci return conversie4to2(numar)
            sfarsit_daca
            daca bazaS==8
                        atunci return conversie8to2(numar)
            sfarsit_daca
            daca bazaS==16
                        atunci return conversie16to2(numar)
            sfarsit_daca
sfarsit_daca
daca bazaS>bazaD // Conversie prin impartiri succesive
      atunci return conversieImpartiri(numar,bazaS,bazaD)
sfarsit_daca
daca bazaS<bazaD // Conversie prin substitutie
     atunci return conversieSubstitutie(numar,bazaS,bazaD)
sfarsit_daca
```

Teste

Utilizator	Aplicație
Validarea datelor de intrare (se introduc date pentru operația de înmulțire)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
3	-
-	Baza:
0	-
-	Dati un numar >1 pentru baza! Baza:
2	-
-	Numar=
16	-
-	Cifrele numarului trebuie sa fie mai mari ca 0 si mai mici strict decat baza 2 Numar=
-5	-
-	Cifrele numarului trebuie sa fie mai mari ca 0 si mai mici strict decat baza 2 Numar=
1001	-
-	Baza:
6	-
-	Numar=
1552	-
-	Baza:
3	-
-	Cifra=
9	-
-	Cifra trebuie sa fie mai mica strict decat baza 3 Cifra=
0	-
-	Introduceti o singura cifra diferita de 0! Cifra=
2	
	Baza in care se va efectua operatia:

2	
	Dati un numar mai mare decat 2 pentru baza! Baza in care se va efectua operatia:
3	
Test pentru operația de adunare (adunarea nu impune restricții)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
1	-
-	Baza:
5	-
-	Numar=
14423	-
-	Baza:
9	-
-	Numar=
1887	-
-	Baza in care se va efectua operatia:
16	-
	14423(5) + 1887(9) = 4D6(16) + 5B0(16) = A86(16)
Teste pentru operația de scădere	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
2	-
-	Baza:
5	-
-	Numar=
14423	-
-	Baza:
9	-
-	Numar=
1887	-
-	Baza in care se va efectua operatia:
16	-
	Nu se poate efectua scaderea. Descazutul

	este mai mic decat scazatorul.
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
2	-
-	Baza:
9	-
-	Numar=
1887	-
-	Baza:
5	-
-	Numar=
14423	-
-	Baza in care se va efectua operatia:
16	-
	1887(9) - 14423(5) = 5B0(16) - 4D6(16) = DA(16)
Test pentru operația de înmulțire (validarea datelor s-a facut la început)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
3	-
-	Baza:
6	-
-	Numar=
1552	-
-	Baza:
3	-
-	Cifra=
2	
	Baza in care se va efectua operatia:
3	
	1552(6) * 2(3) = 1011201(3)
Test pentru operația de împărțire (validarea datelor s-a facut la început)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire

	4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
4	-
-	Baza:
6	-
-	Numar=
1552	-
-	Baza:
3	-
-	Cifra=
2	
	Baza in care se va efectua operatia:
3	
	1552(6) : 2(3) = 21221(3) rest 0(3)
Teste pentru conversie (validarea datelor s-a facut la început)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
5	-
-	Baza:
6	-
-	Numar=
1552	-
-	Baza in care se va efectua operatia:
13	-
-	1552(6) = 26C(13)
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
5	-
-	Baza:
16	-
-	Numar=
1f22f	-
-	Baza in care se va efectua operatia:

7	-
-	1F22F(16) = 1040552(7)
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
5	-
-	Baza:
16	-
-	Numar=
1f22f	-
-	Baza in care se va efectua operatia:
2	-
-	1F22F(16) = 1111100100010111(2)
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
5	-
-	Baza:
2	-
-	Numar=
10010110	-
-	Baza in care se va efectua operatia:
4	-
-	10010110(2) = 2112(4)
Teste pentru conversia utilizând o bază intermediară (validarea datelor s-a facut la început)	
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
6	-
-	Baza:
6	-
-	Numar=

1552	-
-	Baza:
9	-
-	Baza in care se va efectua operatia:
16	-
-	1552(6) = 1AC(16) = 525(9)
-	Operatiile disponibile: 1. Adunare 2. Scadere 3. Inmultire 4. Impartire 5. Conversie simpla 6. Conversie utilizand baza intermediara x. Iesire din aplicatie Alegeti un numar de la 1 la 6:
6	-
-	Baza:
16	-
-	Numar=
1a5b6	-
-	Baza:
4	-
-	Baza in care se va efectua operatia:
2	-
	1A5B6(16) = 1101001011011O110(2) = 122112312(4)