METODE NUMERICE: Tema #1

Termen de predare: 02 APRILIE 2017

Titulari curs: *Florin Pop, George Popescu*

Responsabili Tema: **Sorin N. Ciolofan, David Iancu, George Iordache**

Obiectivele Temei

Obiectivele generale ale acestei teme sunt:

- familiarizarea cu elementele de baza ale programarii in Octave
- utilizarea matricelor si a functiilor asociate
- implementarea unor structuri de date si parcurgerea acestora

Toate problemele de mai jos trebuie implementate in Octave (pentru corectare se va folosi Octave si nu Matlab).

Exercitiul 1 (15 puncte)

Sa se scrie o functie in Octave care face conversia unui numar dintr-o baza in alta. In sistemul de reprezentare pozitionala, baza este numarul de simboluri distincte (cifre, litere) folosite pentru a reprezenta un numar. Spre exemplu sistemul binar foloseste 2 cifre (0 si 1), sistemul zecimal 10 cifre (0, 1, 2, ...9), sistemul hexazecimal (foloseste 16 simboluri, cele 10 cifre plus 6 litere: a,b,c,d,e,f), sistemul in baza 30 foloseste cele 10 cifre plus 20 de litere.
In general, un sir de simboluri $d_1d_2...d_n$ exprimat in baza $b > 1$ reprezinta numarul: $d_1 * b^{n-1} + d_2 * b^{n-2} + ... + d_n * b^0$ unde $0 \leq d_i < b$. De exemplu 101 in baza 2 reprezinta $1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 5$.
Functia va avea obligatoriu antetul:

```
function r = baza(sursa,b1,b2)
  % Intrari:
  % -> sursa: numarul care trebuie transformat si care e reprezentat in baza b1 (
    este pasat ca string)
  % -> b1: baza sursei
  % -> b2: baza rezultat
  %    2<=b1,b2<=30
  % Iesiri:
  % -> r: rezultatul in baza b2
```

Listing 1: functia de conversie

sursa este un sir de simboluri exprimat in baza ***b1*** si trebuie transformat intr-un sir in baza ***b2***. Sirul obtinut, ***r*** (in baza b2) este returnat de catre functie. Programul accepta orice baze cuprinse intre 2 si 30 ($2 \leq b1, b2 \leq 30$).
Exemplu de apel: baza('1041142',6,16) = cb96; baza('1111110000111',2,30)=8t1, etc
Functia va fi plasata intr-un fisier cu numele **baza.m**.

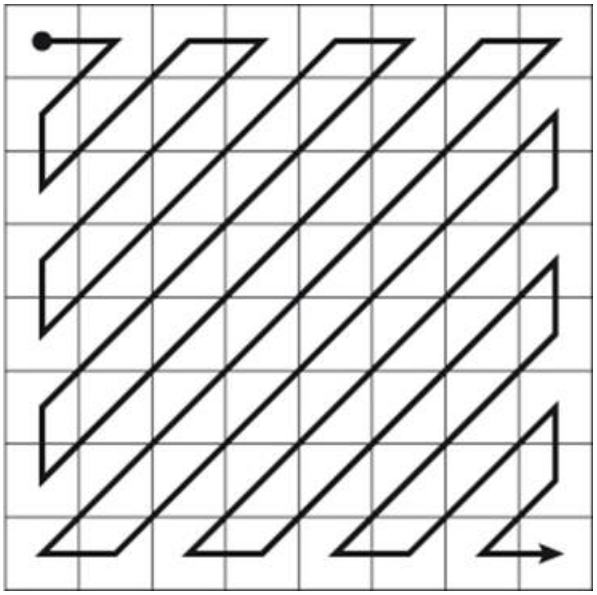

Figure 1: Parcurgerea in zig-zag a unei matrice

Exercitiul 2 (20 puncte)

O matrice zig-zag de ordinul n este o matrice patratica cu n linii si n coloane, in care elementele sunt numere de la 0 la $n^2 - 1$ dispuse in zig-zag, numerele crescind secvential pe masura ce se realizeaza parcurgerea in zig-zag a matricei (sunt parcurse anti-diagonalele matricei, adica acele diagonale care merg din dreapta sus catre stinga jos). Un exemplu de parcurgere in zig-zag este prezentat schematic in Fig.1

De exemplu, matricea zig-zag de ordinul 5 este $Z = \begin{pmatrix} 0 & 1 & 5 & 6 & 14 \\ 2 & 4 & 7 & 13 & 15 \\ 3 & 8 & 12 & 16 & 21 \\ 9 & 11 & 17 & 20 & 22 \\ 10 & 18 & 19 & 23 & 24 \end{pmatrix}$

Sa se scrie o functie in Octave care genereaza matricea zig-zag de ordinul n, unde n este dat ca parametru de intrare functiei. Functia va avea urmatorul antet:

```
function Z = zigzag(n)
  % Intrari:
  % -> n: ordinul matricei zig-zag
  % Iesiri:
  % -> Z, matricea zig-zag
```

Listing 2: functia zigzag

Functia de mai sus va fi plasata intr-un fisier cu numele **zigzag.m**.

Exercitiul 3 (30 puncte)

In codul Morse fiecare litera a alfabetului e codificata printr-o secventa de puncte "." si linii "-". In Fig 2 sunt reprezentate literele din codul Morse sub forma unui arbore. Un punct (.) in cod insemana deplasarea la stinga in arbore iar o linie (-) inseamna deplasarea la dreapta. Se considera ca se porneste din nodul radacina (reprezentat cu galben). Astfel "..." corespunde literei S, "." corespunde lui E iar A este ".-". Prin urmare sirul SEA este codificat in codul Morse ca "... . .-". Observati spatiul care delimiteaza grupurile de "." si "-" care corespund unei litere.

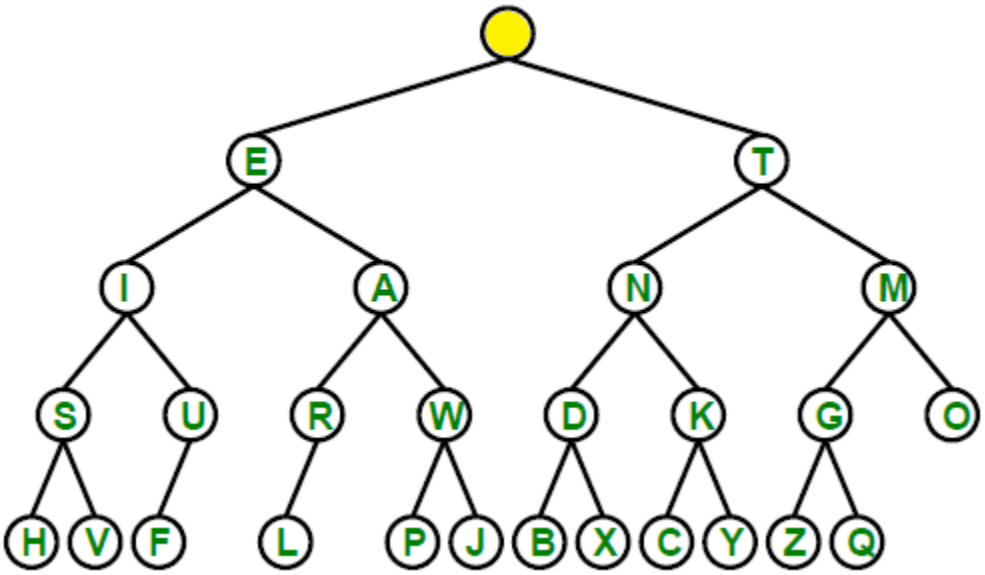

Figure 2: Codul Morse

Pentru a reprezenta in Octave structura de date corespunzatoare codului Morse se vor folosi obligatoriu arbori binari precum cel din Fig.2 reprezentati prin *cell arrays* (tablouri de celule) care sunt tablouri in care elementele pot fi la rindul lor alte structuri de date (inclusiv tablouri de celule).
Pentru mai multe detalii privind cell arrays,
https://www.gnu.org/software/octave/doc/interpreter/Cell-Arrays.html
Observatie! Nu vor fi punctate rezolvarile in care nu se folosesc cell arrays ci alte structuri pentru reprezentarea codului Morse. De exemplu, nu se puncteaza rezolvarile in care se foloseste un vector pentru a stoca fiecare secventa de simboluri "." si "-" corepsunzatoare unei litere.
Codul Morse va fi reprezentat ca un cell array al carui elemente sunt caractere si alte cell arrays. Fiecare celula reprezinta un nod in arbore. Astfel o celula N are trei elemente N{1} reprezentind litera, N{2} alt cell array reprezentind *ramura punct* (ramura la care se ajunge daca se da ".") si N{3} reprezentind *ramura linie* (ramura la care se ajunge daca se da "-"). Exceptiile le reprezinta nodul radacina (care nu are litera ci sirul vid), nodurile R si U care au numai o ramura si frunzele (nodurile terminale, care au zero ramuri).
Pentru construirea arborelui Morse se poate porni cu specificarea celor 12 noduri de pe ultimul nivel, apoi nodurile de pe penultimul nivel, etc, pina se ajunge la nodul radacina:

```
% ultimul nivel
h={'H' {} {}};
v={'V' {} {}};
f={'F' {} {}};
l={'L' {} {}};
p={'P' {} {}};
j={'J' {} {}};
b={'B' {} {}};
x={'X' {} {}};
c={'C' {} {}};
y={'Y' {} {}};
z={'Z' {} {}};
q={'Q' {} {}};

% penultimul nivel
s={'S' h v}
```

```
u={'U' f {}}
r={'R' l {}}
w={'W' p j}
d={'D' b x}
k={'K' c y}
g={'G' z q}
o={'O' {} {}}

% etc... se completeaza celelalte nivele
% ...

M = {'' e t}
```

Listing 3: constructie arbore

1. Sa se scrie o functie care construieste arborele corespunzator codului Morse folosind cell arrays, dupa modelul din Listingul 3 (2.5 puncte).
2. Sa se scrie o functie care decodifica un caracter reprezentat printr-un sir de "." si "-". Functia returneaza caracterul corespunzator. Indicatie: se va urma calea in arbore corespunzatoare secventei de intrare, daca secventa nu corespunde niciunei litere se returneaza "*" (10 puncte)
3. Sa se scrie o functie care codifica un caracter si returneaza un sir de "." si "-" (functia inversa celei de la punctul 2). Indicatie: se va cauta caracterul in arbore, daca nu e gasit se returneaza "*" (15 puncte)
4. Sa se scrie cite o functie care codifica/decodifica mai multe caractere o data, folosind functiile de la punctele 2) si 3) (2.5 puncte).

Functiile de la punctele 1-4 vor avea antetul de mai jos:

```
function m = morse()
  % Intrari:
  % -> nu exista
  % Iesiri:
  % -> m, cell array reprezentind codul morse


function x = morse_decode(sir)
  % Intrari:
  % -> sir: sirul de "." si "-"
  % Iesiri:
  % -> x: caracter sau '*'

function x = morse_encode(c)
  % Intrari:
  % -> c: caracter
  % Iesiri:
  % -> x: sirul de "." si "-"

function x = multiple_decode(sir)
  % Intrari:
  % -> sir: sirul de "." si "-" separat prin spatii
  % Iesiri:
  % -> x: sir de caractere

function x = multiple_encode(str)
  % Intrari:
  % -> str: sir de caractere
  % Iesiri:
```

```
30  % -> x: sirul de "." si "-" separat prin spatii
```

Listing 4: cod Morse

Fiecare din cele 5 functii va fi plasata intr-un fisier propriu: **morse.m**, **morse_encode.m**, **morse_decode.m**, **multiple_encode.m**, **multiple_decode.m**

Exercitiul 4 (25 puncte)

Sa se implementeze jocul X si O pe o tabla 3x3. Implementarea va avea obligatoriu urmatoarele functionalitati:

- utilizatorul poate alege la inceputul fiecarei partide daca vrea sa joace cu X sau cu O
- utilizatorul va putea introduce mutarea sa
- tabla reflectind mutarile jucatorilor va fi afisata in permanenta (fie la stdout fie intr-o fereastra GUI)
- se va pastra scorul partidelor utilizator-calculator. Scorul va fi afisat dupa incheierea fiecarei partide
- va exista o modalitate prin care utilizatorul poate incheia programul (de ex. apasarea tastei 'q', etc).

Obligatoriu va exista o functie Octave cu urmatorul antet

```
1  function [] = joc()
2    % nu are parametri de intrare sau iesire
```

Functia de mai sus va fi plasata intr-un fisier cu numele **joc.m**
Detaliile de implementare raman la latitudinea studentilor.

Detalii de implementare si redactare (de citit de mai multe ori cu atentie!)

Pentru implementarea temei, puteti scrie in plus fata de ce se cere si alte functii ajutatoare, dar cele solicitate in enuntul exercitiilor trebuie obligatoriu sa existe conform antetului specificat. Orice nerespectare oricit de minora a antetului (nume schimbat al functiei chiar si cu o litera, parametru extra, etc) poate duce la pierderea punctajului pentru acea functie (deoarece scriptul de corectare va esua). Mai mult, trebuie sa tineti cont de urmatoarele aspecte:

- ATENTIE: Programele scrise trebuie sa functioneze pentru orice valori (valide bineinteles) ale inputului. Temele vor fi testate si pentru date de intrare care nu au fost comunicate anterior, pentru a descuraja implementarea de "solutii" care functioneaza doar pentru anumite date de intrare.
- Codul sursa va contine comentarii semnificative si sugestive cu privire la implementarea cerintelor;
- Existenta unui fisier `README` care va prezenta detaliile legate de implementarea temei;
- Fisierele sursa *.m folosite in rezolvarea temei impreuna cu fisierul `README` vor fi incluse intr-o arhiva `.zip`. Denumirea arhivei respecta specificatiile din regulamentul cursului;
- Tema se va implementa obligatoriu in Octave.
- Se acorda 90 de puncte pentru o tema care functioneaza conform cerintelor descrise mai sus. In plus, 10 puncte se pot obtine pentru comentariile din fisierele sursa si fisierul `README`.