Introducere în programare, 2016-2017

Tema 2

Exerciţiul 1. Reprezentare prin ADN

Un informatician a descoperit un nou mod de a codifica un număr întreg folosind acidul dezoxiribonucleic, ce conține A (adenină), C (citozină), G (guanină) și T (timină).
Astfel, în calcule se folosește baza de numerație 4, dar cifrele A, C, G și T au valorile A=0, C=1, G=2, iar T=-1.
De exemplu, numărul întreg 27 se reprezintă, folosind ADN, prin "GTT". Asemănător, numărul întreg -6 se reprezintă prin "TGG", iar numărul 84 se reprezintă prin "CCCA".

Justificare:

$GTT = Gx4^2+Tx4^1+Tx4^0 = 2x16+(-1)x4+(-1)x1 = 32-4-1=27$
$TGG = Tx4^2+Gx4^1+Gx4^0 = (-1)x16+2x4+2x1 = -16+8+2=-6$
$CCCA = Cx4^3+Cx4^2+Cx4^1+Ax4^0=1x64+1x16+1x4+0x1=84$

Se consideră declarația

```
#define MAX_ADN 35
typedef char sirADN[MAX_ADN];
```

Cerinţele exerciţiului:

a. (1p) Scrieți o funcție

```
void codificareADN(long nrIntreg, sirADN codADN)
```

care codifica un număr întreg `nrIntreg` în reprezentarea ADN.

De exemplu, pentru `nrIntreg=27`*, funcția va seta pe* `sirADN` *la* "GTT".

b. (1p) Scrieți o funcție

```
long decodificareADN(char sirADN[MAX_ADN])
```

care decodifica un șir ADN într-un număr întreg, de tip `long`.

De exemplu, pentru `sirADN="GTT"` *funcția va returna* `27`.

c. (2p) Scrieți o funcție

```
void adunareADN(sirADN primulSir, sirADN alDoileaSir, sirADN suma)
```

care adună două șiruri ADN, rezultatul fiind reprezentarea în ADN a sumei celor două numere, reprezentate de `primulSir` și `alDoileaSir`.

De exemplu, pentru `primulSir="GTT"` *și* `alDoileaSir="C"`*, funcţia va obține* `suma="GTA"`.

d. (2p) Scrieți o funcție

```
void scadereADN(sirADN primulSir, sirADN alDoileaSir, sirADN suma)
```

care scade două șiruri ADN, rezultatul fiind reprezentarea în ADN a sumei celor două numere, reprezentate de `primulSir` și `alDoileaSir`.

De exemplu, pentru `primulSir="GTA"` *și* `alDoileaSir="C"`*, funcţia va obține* `suma="GTT"`.

Exerciţiul 2. O fată şi un băiat

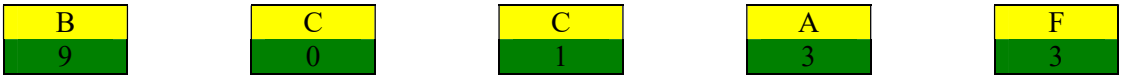
O fată şi un băiat joacă următorul joc de echipă.

Fata dispune de o serie de cartonașe dreptunghiulare, în care, în partea de sus apare o literă mare a alfabetului englezesc (‘A’…’Z’), iar în partea de jos apare una sau apar două litere ale alfabetului englezesc (‘A’..’Z’).

Exemple de cartonaşe pe care le poate avea fata:

Băiatul dispune de o serie de cartonașe dreptunghiulare, în care, în partea de sus apare o literă mare a alfabetului englezesc (‘A’…’Z’), iar în partea de jos apare o cifră (‘0’..’9’).

Exemple de cartonaşe pe care le poate avea băiatul:



Separat, într-o cutie, există o mulţime suficient de mare de cartonaşe pătrate, mici, pe care sunt scrise atât cifrele de la ‘0’ la ‘9’, cât şi literele de la ‘A’ la ‘Z’ şi care se pot repeta de mai multe ori. Pentru derularea jocului, există un număr suficient de cartonaşe cu fiecare cifră sau literă.

La începutul jocului, cei doi iau din cutie mai multe cartonaşe cu cifre, formând un şir de cifre, numit **T**. (**T** are cel puţin o cifră).

*Exemple de şiruri **T**:*

Jocul constă în obţinerea, de către fată şi băiat (împreună) a acestui şir de cifre **T**, folosind cartonaşele date şi plecând de la un şir **S**, care iniţial este format dintr-o singură literă (de exemplu, **S** este “A” sau **S** este “T”).

Cei doi jucători fac mutări, dar NU neapărat alternativ, ci de comun acord, cu scopul de a obține T, plecând de la S, folosind următoarele proceduri.

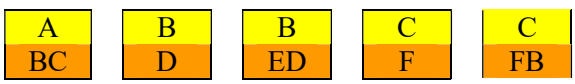
Fata poate înlocui în **S** orice literă pe care o are pe un cartonaș în zona galbenă cu cele două litere pe care le are în zona portocalie. Ea poate face acest lucru de 0, 1 sau mai multe ori, refolosind cartonaşele,

Băiatul poate înlocui în **S** orice literă pe care o are pe un cartonaș în zona galbenă cu cifra pe care o are în zona verde. El poate face acest lucru de 0, 1 sau mai multe ori, refolosind cartonaşele.

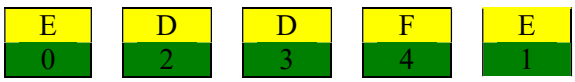
Scopul lor este de a obține pe **T**, plecând de la **S**, folosind orice mutare de înlocuire posibilă, de oricâte ori este nevoie.

Exemplu de joc:

Fata dispune de:



Băiatul dispune de:



Arbitrul stabilește șirul de oprire **T** ca fiind

pornire **S** ca fiind “A”, deci:

Iată cum pot juca cei doi jucători, pentru a obține pe T, plecând de la S:

1. Mai întâi fata folosește cartonașul (A,BC) și înlocuiește pe A cu BC, obținând:

2. Apoi, tot ea folosește cartonașul (B,D) și obține:

3. Acum intervine băiatul și folosește cartonașul său (D,2), obținându-se:

4. Fata vine acum cu folosirea cartonașului (C,FB) și obține:

5. Acum băiatul folosește cartonașul (F,4) și se obține:

6. Fata folosește cartonașul (B,ED) și obține șirul:

7. Acum băiatul face două mutări la rând, întâi folosind cartonașul (E,0), obținând:

8. iar apoi cartonașul (D,3), obținând șirul **T** final:

Cerința exercițiului:

Se consideră următoarele declarații:

```
#define MAX_CARTONASE 1000
#define MAX_SIR 15

typedef struct {
    char galben;
    union {
        char verde;
        char portocaliu[3];
    };
} cartonas;

struct mutare {
    cartonas nod;
    struct mutare *urmator;
};

typedef mutare* lista;
```

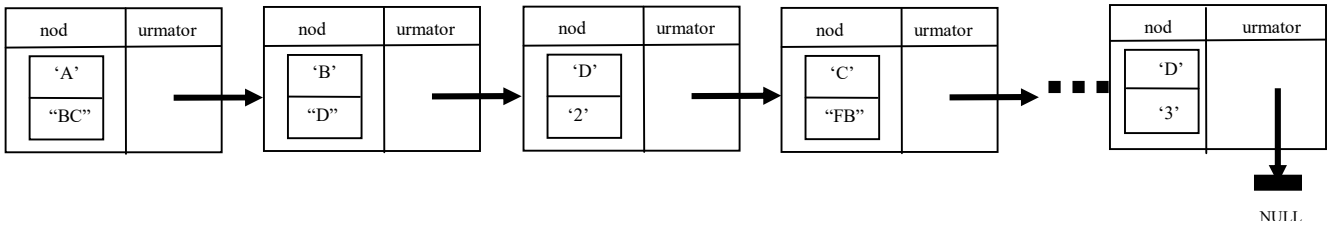
```
struct rezultat {
      bool sePoate;
      int nrMutari;
      lista mutari;
};
```

(5p+2p extra) Să se scrie o funcție care să returneze o structură de tip `rezultat`, care să conțină:

- în câmpul `sePoate`: valoarea true, dacă și numai dacă cei doi jucători pot obține șirul `sirFinal` plecând din șirul format inițial doar din caracterul `literaStart` și folosind cartonașele din vectorul `vecCartonase` (2p), de dimensiune `nCartonase`;
- în câmpul `nrMutari`: numărul minim de mutări ale echipei pentru a obține șirul `sirFinal` plecând din șirul format inițial doar din caracterul `literaStart` și folosind cartonașele din vectorul `vecCartonase`, doar dacă `sePoate` va avea valoarea `true`, respectiv 0 dacă `sePoate` va avea valoarea `false` (3p);
- în câmpul `mutari`: o listă simplu înlănţuită, de lungime minimă, în care nodurile conţin mutările echipei (cartonaşele folosite), dacă și numai dacă cei doi jucători pot obține șirul `sirFinal` plecând din șirul format inițial doar din caracterul `literaStart` și folosind cartonașele din vectorul `vecCartonase`, respectiv `NULL`, dacă cei doi jucători nu-l pot obține pe `sirFinal` (2p extra).

```
rezultat mutariJoc(cartonas vecCartonase[MAX_CARTONASE], unsigned int
nCartonase, char literaStart, char sirFinal[MAX_SIR])
```

Pentru exemplul ilustrat mai sus, funcţia ar returna lista:



Observaţii:

- Se vor da date corecte de test și care să nu depășească dimensiunile de stocare necesare, pentru fiecare problemă în parte.
- Se pot folosi funcţiile din `<string>` şi din `<cstring>` (`<string.h>`).
- Pentru problema 2 se poate poate implementa algoritmul Cocke-Younger-Kasami (CYK sau CKY).