Indicații pentru Tema 8

- **Tema 8** va avea **11 probleme**, fiecare punctată cu note de la 1 la 10 astfel:
  - Indentare corespunzătoare = **1 puncte**
  - Utilizarea acoladelor, lizibilitatea codului = **1 puncte**
  - Utilizarea corespunzătoare a instrucțiunilor de tip **if**, **for**, **while**, **do...while** = **1 punct**
  - Declararea corespunzătoare de variabile = **1 punct**
  - Funcționalitatea corespunzătoare a programului (adică programul face ce trebuie să facă) = **2 puncte**
  - Utilizarea corespunzătoare a pointerilor = **2 puncte**
  - Utilizarea corespunzătoare a alocării dinamice = **2 puncte**

- **Tema trebuie încărcată pe Campus Virtual până cel târziu MIERCURI, 11 decembrie 2019, ora 23:59 (hard deadline).**

- **TOATE PROBLEMELE SUNT OBLIGATORII!!**

- **Vor fi încărcate 11 fișiere cu extensia .c denumite astfel: p1.c, p2.c, p3.c, p4.c, p5.c, p6.c, p7.c, p8.c, p9.c, p10.c, și p11.c corespunzătoare rezolvărilor problemelor 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, respectiv 11.**

- **NU încărcați proiecte CodeBlocks!!! Numai fișierele cu extensia .c denumite corespunzător dupa cele menționate la subpunctul anterior.**

- **IMPORTANT!! Dacă nu compilează codul, problema este notată cu 0. Punctul din oficiu pe fiecare problema se acordă în cazul în care codul compilează și nu are nicio eroare de compilare!**

- **Tema este OBLIGATORIE și cei care nu o fac nu vor putea da Testul 3!!**

- **Copiatul este strict interzis! Dacă sunt găsite astfel de cazuri, toți studenții implicați (adică cei care copiază și cei de la care se copiază) vor fi exmatriculați.**

- **Încercați să rezolvați voi singuri problemele. Dacă aveți nevoie de ajutor, vă stau la dispoziție.**

- **Pentru orice fel de întrebări nu ezitați să mă contactați.**

Tema 8

Problema 1

Alocați dinamic o semimatrice pătratică de tip întreg. Prin semimatrice se înțelege partea de deasupra diagonalei secundare, inclusiv diagonala secundară. Dimensiunea matricii, n, se va citi de la tastatură. Inițializați semimatricea cu valori după următoarele reguli: elementele de pe prima linie și prima coloană sunt 1, iar restul elementelor sunt inițializate după formula `a[i][j]=a[i-1][j]+a[i][j-1]`. **Trebuie utilizați pointeri și alocare dinamică.**

Exemplu: pentru n=5 matricea va fi: $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & \\ 1 & 3 & 6 & & \\ 1 & 4 & & & \\ 1 & & & & \end{pmatrix}$

Problema 2

Se citesc de la tastatură cuvinte până la introducerea cuvântului `"gata"` (acesta din urmă **nu** va fi prelucrat). Păstrați cuvintele într-un vector de șiruri de caractere. Vectorul și fiecare șir de caractere vor fi alocate dinamic. Pentru fiecare cuvânt găsiți și afișați perechea cea mai potrivită. Perechea cea mai potrivită este acel cuvânt cu proprietatea că un număr maxim de litere din alfabet apare în ambele cuvinte. Nu se face distincție între literele mari și literele mici. **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: perechea de cuvinte SOARE și cerebel apar comune două litere din alfabet: e și r.

Problema 3

Să se citească un text format din mai multe fraze, câte una pe fiecare linie. Textul se termină cu linie vidă. Să se memoreze textul într-o zonă alocată dinamic și să se afișeze textul. Scrieți o funcție care caută în text toate aparițiile unui șir de caractere citit de la tastatură și îl înlocuiește cu un alt șir citit de la tastatură. Programul principal va trebui să afișeze textul modificat pe ecran. **Trebuie utilizați pointeri și alocare dinamică pentru toate șirurile de caractere, vectori și matrici. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citește textul

```
"Ana are mere.
Maria are mere."
```

Programul principal va afișa textul de mai jos dacă va fi citit de la tastatură șirul de caractere `"mere"` și șirul de caractere `"pere"`:

```
"Ana are pere.
Maria are pere."
```

Problema 4

Să se implementeze o funcție `char *duplicat(const char *sursa)`. Funcția va crea o dublură a șirului `sursa`, adică va aloca dinamic memoria necesară și va copia șirul `sursa`. Citiți n cuvinte de dimensiune variabilă de la tastatură și memorați-le într-un vector folosind funcția duplicat. Să se afișeze doar cuvintele ușoare din vector utilizând o funcție separată. Cuvintele ușoare sunt acele cuvinte care au mai mult de 2/3 din litere vocale (consoanele sunt strict mai puțin de 1/3). **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citesc de la tastatură n=2 cuvinte `"aer"` și `"aeroplanare"` va fi afișat pe ecran cuvântul `"aer"`.

Problema 5

Un editor T9 modificat poate introduce doar cifre cărora le corespund litere: cifrei 2 îi corespund literele abc, cifrei 3 def, cifrei 4 ghi, cifrei 5 jkl, cifrei 6 mno, cifrei 7 pqrs, cifrei 8 tuv, cifrei 9 wxyz. Cifra 0 va fi un spațiu. Se va citi de la tastatură un text alocat dinamic format din numere și cifre despărțite prin spații. Va fi utilizat un dicționar sub formă de vector alocat dinamic pentru traducerea cifrelor în litere într-o funcție separată. Această funcție va fi apelată astfel încât după citirea textului din numere și cifre va fi afișat pe ecran mesajul decodificat. **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citește de la tastatură 666 55 se va afisa mesajul `"ok"`. Dacă se citeste de la tastatură 22 33 6 0 2 9999 444 se va afisa mesajul `"bem azi"`.

Problema 6

Să se scrie funcția `int anagrama(char *s1, char *s2)` care primește ca parametri două șiruri de caractere și decide dacă șirul `s2` este o anagramă a șirului `s1`. Dacă este o anagramă va returna valoarea 1, altfel va returna valoarea 0. Un cuvânt este o anagramă a altui cuvânt dacă conține exact aceleași litere ca primul cuvânt, dar în altă ordine. Citiți de la tastatură mai multe cuvinte separate prin spații pînă la întâlnirea cuvântului `"gata"`. Cuvintele se vor păstra în memorie. Dacă se constată că un cuvânt citit este o anagramă a unuia din cuvintele deja păstrate, atunci noul cuvânt nu va mai fi păstrat în memorie. În final vor fi afișate toate cuvintele memorate. Utilizați spațiul de memorie minim necesar. **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citesc cuvintele `"ana naa eva ave mara rama gata"` vor fi memorate numai cuvintele `"ana"`, `"eva"` și `"mara"`, iar pe ecran vor fi afișate: `"ana eva mara"`

Problema 7

Să se citească de la tastatură un text format din mai multe linii, terminat cu linie vidă. Să se afișeze cu ajutorul unei funcții liniile textului în ordinea descrescătoare a lungimii lor. Pentru memorarea datelor se va folosi spațiul de memorie minim necesar. **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citeste textul:

```
"Ana are mere.
Ion iubeste pamantul.
Moromete nu se intelege cu familia sa."
```

Se va afișa pe ecran:

```
"Moromete nu se intelege cu familia sa.
Ion iubeste pamantul.
Ana are mere."
```

Problema 8

Să se citească de la tastatură un text alocat dinamic până la întâlnirea cuvântului `"gata"`. Împărțiți cuvintele din text în silabe după regula simplă că între două consoane alăturate se introduce caracterul `'-'`. Afișați pe ecran textul rezultat. **Trebuie utilizați pointeri și alocare dinamică. Puteți încerca să utilizați iterarea cu pointeri.**

Exemplu: dacă se citește textul `"doamne da sfantul cinci la programare"` se va afișa pe ecran `"doam-ne da s-fan-tul cin-ci la p-rog-ramare"`

Problema 9

Să se citească de la tastatură o matrice de m linii și n coloane cu valori reale. Memorați matricea într-o zonă de memorie alocată dinamic. Să se genereze și să se afișeze un șir care să conțină acele elemente care sunt mai mare decât media aritmetică a elementelor vecine. Elementele vecine sunt cele aflate deasupra, dedesubt, la stânga sau la dreapta, deci un element poate avea maxim 4 vecini. Pentru generarea șirului va fi folosită o funcție, iar pentru afișarea șirului va fi utilizată o altă funcție. **Trebuie utilizați pointeri și alocare dinamică.**

Exemplu: se citește de la tastatură m=3, n=2 și matricea: $\begin{pmatrix} 1 & 2 \\ 10 & 4 \\ 5 & 6 \end{pmatrix}$. Șirul generat va fi: 10 și 6, iar pe ecran vor fi afișate nuumerele 10 și 6.

Problema 10

Să se citească o matrice de m linii și n coloane cu numere întregi fără semn. Scrieți o funcție care introduce o nouă coloană în matrice, elementul de pe fiecare linie reprezentând suma biților de 1 din reprezentările interne ale fiecărui element de pe acea linie a matricii. În programul principal apelați funcția creată și afișați matricea nouă pe ecran. Pentru memorarea numerelor și orice alte prelucrări utilizați spațiul de memorie minim necesar. **Trebuie utilizați pointeri și alocare dinamică.**

Exemplu: dacă se citește de la tastatură m=3, n=2 și matricea: $\begin{pmatrix} 1 & 2 \\ 10 & 4 \\ 5 & 6 \end{pmatrix}$ atunci vom avea afișată matricea $\begin{pmatrix} 1 & 2 & 2 \\ 10 & 4 & 3 \\ 5 & 6 & 4 \end{pmatrix}$.

Problema 11

Să se citească de la tastatură o **matrice de caractere** de m linii și n coloane, m și n citite de la tastatură în prealabil. Matricea reprezintă un text în care cuvintele sunt separate de spații sau semne de punctuație (semnele de punctuație luate în considerare sunt `-,;?!"'`. De asemenea, cuvintele pot fi despărțite în silabe la sfârșit de rând. Scrieți o funcție care reconstituie textul conținut în matrice și să se păstreze într-un șir de caractere, unind cuvintele despărțite în silabe. Afișați în programul principal șirul de caractere construit. Utilizați spațiul de memorie minim necesar. Observație: citiți **matricea de caractere** ca o matrice normală. **Trebuie utilizați pointeri și alocare dinamică**.

Exemplu: dacă se citește de la tastatură m=3, n=2 și matricea: $\begin{pmatrix} 'a' & 'n' & - \\ 'd' & 'a' & ' ' \\ 'a' & 't' & '.' \end{pmatrix}$ șirul care trebuie afișat pe ecran este `"anda at."`