PROGRAMAREA CALCULATOARELOR ȘI LIMBAJE DE PROGRAMARE I
Tema #1 Funcții, Vectori și Matrici

Deadline soft: 10.11.2024 23:55. Deadline hard: 13.11.2024 23:55

Coordonator: Darius Neațu

Responsabili temă: Mihai-Cătălin Stan, Raluca-Monica Bîrlădeanu, Denis Covei, Andrei Stan

Responsabil checker: Mihai-Cătălin Stan, Raluca-Monica Bîrlădeanu, Andrei Stan

Changelog:

- 28.10.2024: Publicare tema 1.
  Vă rugăm să adresați toate întrebările legate de teme pe forumul asociat temei 1 de pe site-ul de curs. Orice clarificare / corectare / actualizare legată de enunț, checker sau lucruri administrative etc, vor fi anunțate exclusiv pe forumul dedicat temei 1.
- 1.11.2024: Publicare checker.
  - Modificare formatare output problema 1. Output-ul a fost restrâns prin eliminarea șirurilor de caractere repetitive.
  - Modificare restricții problema 2.
  - Remediere exemple problema 3. Am corectat numărul de fire în primul exemplu și ordinea parametrilor din output în cel de al doilea exemplu.
  - Clarificare date de ieșire problema 4. Am clarificat modul în care se pun caracterele de tip newline.
- 2.11.2024: Clarificări restricții și date de ieșire pentru problema 3.
- 4.11.2024: Corectare nume fișiere sursă.

Contents

Problema 1 - Jocuri de societate

Enunț

Gigel este rezident în Regie în anul I. Acesta este pasionat de jocuri de societate și de curând și-a găsit un grup de prieteni cu care împărtășește această pasiune. Din păcate, Gigel este cel mai slab din grup, așa că s-a decis să își dezvolte o strategie pe care să o testeze folosind un program în C care simulează o partidă de joc folosind strategia pe care a dezvoltat-o.

În jocul acesta, caracterul lui Gigel are un hp (health points/puncte de viață) dat. Acesta poate primi până la **N** obiecte și se va lupta cu **M** inamici. Scopul jocului este să supraviețuiască la cât mai multe bătălii folosindu-se eficient de obiectele pe care le are. Obiectele pot fi de Heal sau de Shield. Cele de Heal îl vor ajuta pe Gigel să își crească hp-ul, în timp ce cele de Shield vor scădea daunele provocate de inamici.

Strategia pe care a adoptat-o este următoarea:

1. Toate obiectele de heal vor fi folosite înainte de primul bossfight (un bossfight este o luptă cu un inamic - considerăm că inamicul poate cauza daune egale cu un input asociat acestuia) și vor fi adunate cumulativ la hp-ul jucătorului.
2. Inamicii sunt întâlniți în ordinea dată de datele de intrare. Gigel va folosi cel mai mare item de tip shield, mai mic sau egal cu daunele inamicului curent. Dacă nu vor mai exista scuturi mai mici sau egale cu daunele produse, va folosi cel mai mic scut rămas.

Scrieți un program în C care să primească ca input informații despre joc și să simuleze jocul lui Gigel pe care îl aveți descris mai sus.

Restricții și precizări

- $0 \leq N \leq 10000$
- $0 \leq M \leq 10^7$
- Valoarea unui scut sau valoarea daunei produsă de un bossfight încape pe 32 de biți.
- Se garantează că toate rezultatele intermediare încap pe 64 de biți.
- Atenție! Memoria de pe platforma VMChecker este limitată pentru această problemă. Încercați să folosiți eficient resursele alocate pentru un punctaj complet.

Date de intrare

Toate datele se vor citi de la **STDIN**.

- Pe prima linie se află un număr întreg HP care reprezintă punctele de viață cu care începe caracterul.
- Pe următoarea linie se găsește un număr întreg N care reprezintă numărul total de obiecte pe care le va deține caracterul.
- Pe urmatoarele N linii, perechi de forma **Tip Valoare**, unde Tip poate fi caracterul 'H' (Heal) sau 'S' (Shield), iar Valoare este un număr întreg.
- Pe următoarea linie va fi numărul total de bossfight-uri, M.
- Pe următoarea linie M numere întregi care reprezintă daunele provocate de inamici.

Între oricare doi tokeni vizibili din input există exact un spațiu. La finalul oricărei linii din input există un caracter **newline**.

```
HP
N
Tip_0 Valoare_0
Tip_1 Valoare_1
Tip_2 Valoare_2
...
Tip_{n - 1} Valoare_{n -1}
M
Dmg_0 Dmg_1 ... Dmg_{n - 1}
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**.

- Pe prima linie se va afișa hp-ul inițial, ca în exemplu.
- În continuare, pe următoarele M linii va fi afișat rezultatul luptei cu cei M inamici. Dacă nu este folosit niciun shield sau dacă shield-ul are valoarea nulă, nu se va afișa nimic în locul acestuia.
- Ultima linie este mereu un mesaj dintre "You died." (în cazul în care caracterul nu a supraviețuit) sau "Foe Vanquished!" (în cazul în care caracterul a înfrânt cu succes toți inamicii).

Între oricare doi tokeni vizibili din output există exact un spațiu. La finalul oricărei linii din output există un caracter **newline**.

```
Initial health points: HP
hp_0 s_0
hp_1 s_1
...
hp_{m - 1} s_{n - 1}
Mesaj_final
```

Exemplul 1

Date de intrare

```
10
3
S 20
H 15
S 10
3
30 15 25
```

Date de ieșire

```
Initial health points: 25
15 20
10 10
0
You died.
```

Explicație 1

1. Punctele de viață de la începutul jocului, după aplicare obiectelor de heal va fi 25 (10 Initial Health + 15 Item de heal).
2. În continuare trecem prin lista de inamici, unul câte unul.
3. Întrucât primul inamic cauzează 30 de daune jucătorului, cel mai bun scut pe care îl putem folosi în acest caz conform strategiei de mai sus este cel de 20, fiind cel mai mare scut mai mic decât daunele inamicului. Prin urmare, atacându-ne un inamic cu daune 30 și folosind un scut de 20, la finalul acestei ture personajul va lua doar $30 - 20 = 10$ daune și rămâne cu $25 - 10 = 15$ hp.
4. În cazul celui de al doilea inamic (cu 15 daune), mai avem un singur scut (10), pe care îl vom folosi oricum, fiind ultimul scut rămas. Prin urmare, vom primi $15 - 10 = 5$ daune, și viața caracterului va scădea la $15 - 5 = 10$hp.
5. În cele din urmă, mai avem doar un inamic cu 25 daune și niciun scut rămas, deoarece hp-ul nostru (10) este mai mic decat daunele provocate de inamic (25) și nu mai avem niciun scut rămas, caracterul nostru pierde și se va afișa mesajul "You died.".

Exemplul 2

Date de intrare

```
64
9
S 39
S 91
S 26
H 76
S 38
H 44
H 78
H 22
H 10
4
68 33 16 6
```

Date de ieșire

```
Initial health points: 294
265 39
258 26
258 38
258 91
Foe Vanquished!
```

Explicație 2

1. Punctele de viață după aplicarea tuturor obiectelor de Heal este 294, deci cu atât vom începe jocul.
2. Întrucât cel mai mare shield mai mic decât daunele inamicului curent este 39, vom primi $68 - 39 = 29$ daune. Aceeași regulă se aplică și la cel de-al doilea bossfight.
3. Pentru următoarele 2 bossfight-uri, din moment ce nu mai avem scuturi mai mici decât daunele inamicului curent, vom folosi cel mai mic scut mai mare decât daunele acestuia. În cazul celui de-al treilea bossfight, vom folosi scutul de valoare 38, iar în cazul ultimului vom folosi scutul de valoare 91. În niciunul dintre bossfight-urile menționate nu vom lua daune.
4. La final, se afișează mesajul corespunzător **Foe Vanquished!**.

Problema 2 - Forme

Enunț

Gigelinho, fratele mai mic al lui Gigel tocmai a intrat la liceu și este pasionat de informatică. Gigel, fiind deja student la Calculatoare i-a propus fratelui său o problema ușoară și interactivă. Se da un număr N și N comenzi care cer afișarea a câte uneia din următoarele 5 forme geometrice: pătrat, dreptunghi, triunghi dreptunghic, cruce și fereastră, iar unele forme se pot roti după un unghi dat în sensul acelor de ceasornic.

Ajutați-l pe Gigelinho să scrie un cod în C care să afișeze și rotească aceste forme pe baza unui input primit.

Restricții și precizări

- $1 <= N <= 10^2$
- Oricare dintre dimensiunile citite în datele de intrare (‘<dimensiune>‘ , ‘<latime>‘, ‘<lungime>‘, ‘<lungime_cateta>‘) sunt numere în intervalul $[-50, 50]$.
- Unghiul de rotație specificat în input (‘<unghi>‘) este un număr întreg pe 16 de biți.
- Unghiul de rotație al unui pătrat sau al unei cruci trebuie să fie un multiplu de 45°.
- Unghiul de rotație al unui triunghi dreptunghic trebuie să fie un multiplu de 90°.
- Dimensiunea a unei cruci trebuie să fie un număr impar.
- Dimensiunea a unei ferestre trebuie să fie un număr impar mai mare sau egal ca 5.
- La finalul unui rând, la afișare, nu se vor insera spații adiționale, ci doar caracterul **newline**.

Date de intrare

- Pe prima linie se va citi un număr natural nenul N.
- Pe următoarele N linii, date în următorul format:
  - `p <dimensiune> <unghi>`
    Va desena un pătrat cu latura egală cu <dimensiune>, care este rotit cu unghiul dat în grade.
  - `d <lățime> <înălțime>`
    Va desena un dreptunghi cu lățimea și înălțimea date.
  - `t <lungime_catetă> <unghi>`
    Va desena un triunghi dreptunghic cu lungimea unei catete date, rotit cu unghiul dat în grade.
  - `c <dimensiune> <unghi>`
    Va desena o cruce cu dimensiunea specificată, care este rotită cu unghiul dat în grade.
  - `f <dimensiune>`
    Va desena o fereastră (o cruce în chenarul unui pătrat) de dimensiunea dată.

Toate datele se vor citi de la **STDIN**.

```
N
forma_0 caracteristici_forma_0
forma_1 caracteristici_forma_1
...
forma_i caracteristici_forma_i
forma_{i + 1} caracteristici_forma_{i + 1} caracteristici_forma_{i + 1}
...
forma_{n - 1} caracteristici_forma_{n - 1}
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Se vor afișa formele cerute pe care se aplică rotațiile la unghiurile date dacă este cazul.

- În cazul în care dimensiunea formei nu se încadrează în formatul specificat la intrare, programul va afișa următorul mesaj de eroare: “Unsupported size to display shape”.

- În mod similar, dacă unghiul dat la intrare este invalid (adică nu este unul dintre unghiurile suportate), programul va afișa următorul mesaj de eroare: "Unsupported angle to display shape".
- În cazul în care dimensiunea formei și unghiul de rotație sunt simultan invalide, se va afișa un singur mesaj de eroare, anume: "Unsupported size to display shape".

```
forma_0
forma_1
...
forma_n
```

Între oricare doi tokeni vizibili din input există exact un spațiu. La finalul oricărei linii din input și din output există un caracter **newline**. Între fiecare două forme afișate, va fi adăugat încă un caracter **newline**.

Exemplul 1

Date de intrare

```
12
p 3 -90
p -100 45
p 4 45
d 7 5
t 3 0
t 3 90
t 3 180
t 3 270
t 3 12
c 5 90
c 5 135
f 7
```

Date de ieșire

```
***
***
***

Unsupported size to display shape

    *
   ***
  *****
*******
  *****
   ***
    *

*******
*******
*******
*******
*******

*
**
***
```

```
24
25 ***
26 **
27 *
28
29 ***
30  **
31   *
32
33   *
34  **
35 ***
36
37 Unsupported angle to display shape
38
39   *
40   *
41 *****
42   *
43   *
44
45 *   *
46  * *
47   *
48  * *
49 *   *
50
51 *******
52 *  *  *
53 *  *  *
54 *******
55 *  *  *
56 *  *  *
57 *******
```

Explicație

- Prima comandă cere afișarea unui pătrat de latură 3 (3 steluțe vor fi afișate pe fiecare latură) și va fi rotit la -90 de grade, un unghi valid fiind divizibil cu 45, din care va rezulta tot un pătrat normal.
- A doua comandă oferă un input invalid, din moment ce nu putem avea o latură de dimensiune negativă, și se va afișa un mesaj de eroare corespunzător.
- A treia comandă cere afișarea unui pătrat cu latura 4 rotit la 45 de grade, deci va fi afișat asemeni unui romb.
- A 4-a comandă cere afișarea unui dreptunghi de lățime 7 (7 coloane) și înălțime 5 (5 linii).
- Următoarele 4 comenzi afișeaza un triunghi dreptunghic în toate modurile în care acesta poate fi rotit.
- Comanda "t 3 12" este una invalidă folosind un unghi de rotatie de 12 grade, un numar ce nu este divizibil cu 45.
- Următoarele 2 comenzi prezintă modul de afișare a unei cruci și a unghiului de rotație în care se va afișa diferit.
- Ultima comandă prezintă cum este structurată o fereastră de dimensiune 7 în output.

Problema 3 - Simulator de circuite

Enunț

Gigel este student în anul 2 la Calculatoare, dar a rămas restant la Electrotehnică - o materie din anul I semestrul 2 - fiindcă nu știa legile lui Kirchhoff. Pentru a repeta, Gigel s-a decis să scrie un simulator de circuite care să verifice cele două legi.

Legea I a lui Kirchhoff: Suma intensităților curenților care intră într-un nod este egală cu suma intensităților curenților care ies din acest nod.

$$\sum_{k_{in}=1}^{N} I_{in_k} = \sum_{k_{out}=1}^{M} I_{out_k} \tag{1}$$

Această lege mai poate fi scrisă formal și sub forma:

$$\sum_{k=1}^{N} I_k = 0 \tag{2}$$

Pentru ușurință, nodurile vor fi date și vom considera că pot exista noduri ce reprezintă intersecția a doar două fire.

Legea a II-a a lui Kirchhoff: Suma algebrică a tensiunilor de-a lungul oricărui ochi de circuit este nulă.

$$\sum_{i=1}^{N} V_{consumatori} = \sum_{j=1}^{M} V_{surse} \tag{3}$$

Și această lege poate fi scrisă formal sub forma următoare:

$$\sum_{k=1}^{N} V_k = 0 \tag{4}$$

Pentru ușurință, exemplele vor avea mereu doar un ochi de circuit.

Scrieți un program C care să citească legea ce trebuie validată, N noduri și W fire (wires), care va verifica corectitudinea circuitelor date și va afișa un mesaj pe ecran.

Restricții și precizări

- $1 \leq N, W \leq 1000$;
- Componentele sunt doar rezistori - marcati in input de caracterul 'R' - sau surse de tensiune - marcate in input de caracterul 'E'.
- Tensiunile și curenții sunt numere reale pe 32 de biți.
- Se garantează că circuitele pentru Kirchhoff II vor avea un singur ochi de circuit, fără alte bucle. Prin bucle ne referim la situația în care un fir pleacă dintr-un nod, și prin orice cale se întoarce tot în acesta.
- Se garantează că sensul curentului prin fire este dinspre nod_inceput și către nod_final.
- Un circuit poate fi închis sau deschis. Prin circuit deschis ne referim la situația în care un fir pleacă dintr-un nod și niciun alt fir nu intră în acesta, sau vice-versa.
- Numerotarea nodurilor începe de la 1.
- Pentru verificarea legilor, se va verifica rezultatul cu o precizie de 1e-8.

Date de intrare pentru Kirchhoff I

Toate datele se vor citi de la **STDIN**.

- Pe prima linie se va citi caracterul 'I' sau succesiunea de caractere "II" reprezentând legea ce trebuie probată.
- Pe următoarea linie un număr întreg N reprezentând numărul de noduri din circuit.
- Un număr W reprezentând numărul de fire din circuit.
- W perechi de forma **nod_început nod_final curentul_prin_fir** reprezentând curentul care circulă între nodurile date, pe direcția $nod_{inceput} => nod_{final}$.

```
L
N
W
nod_inceput_0 nod_final_0 Curent_0_0
nod_inceput_1 nod_final_1 Curent_1_1
...
nod_inceput_{n - 1} nod_final_{n - 1} Curent_{n - 1}_{n - 1}
```

Date de intrare pentru Kirchhoff II

- Pe prima linie se va citi caracterul 'I' sau succesiunea de caractere "II" reprezentând legea ce trebuie probată.
- Pe următoarea linie un număr întreg N reprezentând numărul de noduri din circuit.
- Un număr W reprezentând numărul de fire din circuit.
- W perechi de forma **nod_început nod_final curentul_prin_fir C**, urmat de C perechi de forma **Componentă Valoare**, reprezentând componentele prezente pe firul dintre $nod_{inceput} => nod_{final}$ și valorile lor.

```
L
N
W
nod_inceput_0 nod_final_0 Curent_0_0 Numar_componente_0_0 T_0 V_0 ... T_{k - 1} V_{k -
    1}
nod_inceput_1 nod_final_1 Curent_1_1 Numar_componente_1_1 T_0 V_0 ... T_{k - 1} V_{k -
    1}
...
nod_inceput_{n - 1} nod_final_{n - 1} Curent_{n - 1}_{n - 1} Numar_componente_{n - 1}_
   {n - 1} T_0 V_0 ... T_{k - 1} V_{k - 1}
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Programul va afișa un mesaj corespunzător în funcție de inputul primit.

- Dacă legea L (I sau II) a lui Kirchhoff se verifică, se va afișa un mesaj de forma "Legea L a lui Kirchhoff se respectă pentru circuitul dat.", unde L este 1 sau 2 in mesajul afișat.
- Dacă prima lege nu se verifcă, se afișează un mesaj de forma "Legea 1 a lui Kirchhoff nu se respecta pentru egalitatea xA = yA in nodul z.", unde x este suma curenților care intră în nodul z, iar y este suma curenților care ies din nodul z. Acest mesaj se va afișa doar pentru primul nod în care legea nu se respectă.
- Dacă a doua lege nu se verifică, se afișează un mesaj de forma "Legea a 2-a a lui Kirchhoff nu se respecta pentru egalitatea xV = yV.", unde x este căderea de tensiune pe componente, iar y este căderea de tensiune pe surse.

- Doar circuitele închise pot fi valide pentru a aplica legile menționate. Dacă se detectează că circuitul este deschis, se va afișa mesajul "Circuitul este deschis in nodul n.", unde n este nodul în care circuitul este întrerupt. Se va afișa mesajul doar pentru primul nod în care circuitul este deschis.
- Tensiunile și curenții trebuie să fie pozitive pentru ca un circuit să fie valid. Dacă inputul unei componente este invalid, se va afișa un mesaj de forma "Componenta dorita nu exista." sau "Piesele nu pot avea valori negative.".
- Tensiunile pentru legea Kirchhoff II trebuie afișate cu exact 9 zecimale în cazul în care legea nu respectă.

Între oricare doi tokeni vizibili din input și output există exact un spațiu. La finalul oricărei linii din input și există un caracter newline.

Exemplul 1

Date de intrare

```
I
6
7
1 2 1
2 3 0.2
2 5 0.8
3 4 0.2
4 5 0.2
5 6 1
6 1 1
```

Date de ieșire

```
Legea 1 a lui Kirchhoff se respecta pentru circuitul dat.
```

Explicație 1

Se va verifica prima lege a lui Kirchhoff într-un circuit cu 6 noduri și 8 fire.

```
1 --- 1A --> 2 --- 0.2A --> 3
^             |              |
|             |              |
1A          0.8A           0.2A
|             |              |
|             V              V
6 <-- 1A --- 5 <-- 0.2A --- 4
```

Următoarele 8 linii din input formează un circuit precum cel de mai sus în care curenții care intră în fiecare nod sunt egali cu cei care ies. Spre exemplu, pentru nodul 1 avem un curent de 1A care iese către nodul 2 și un curent de 1A care intră din nodul 6, $1A = 1A \Rightarrow$ egalitatea se respectă. Pentru nodul 2, avem un curent de 1A care intră din nodul 1, si doi curenți care ies din acest nod, unul de 0.2A către nodul 3 și unul de 0.8A către nodul 5, deci $1A = 0.2A + 0.8A \Rightarrow 1A = 1A$, deci egalitatea se respectă și pentru acest nod. Repetând procesul pentru fiecare nod, observăm că circuitul respectă egalitatea legii Kirchhoff I.

Exemplul 2

Date de intrare

```
II
3
3
1 2 0.5 1 R 10.0
2 3 0.5 1 E 5.0
3 1 0.5 1 R 20.0
```

Date de ieșire

```
Legea a 2-a lui Kirchhoff nu se respecta pentru egalitatea 15.000000000V = 5.000000000
    V.
```

Explicație 2

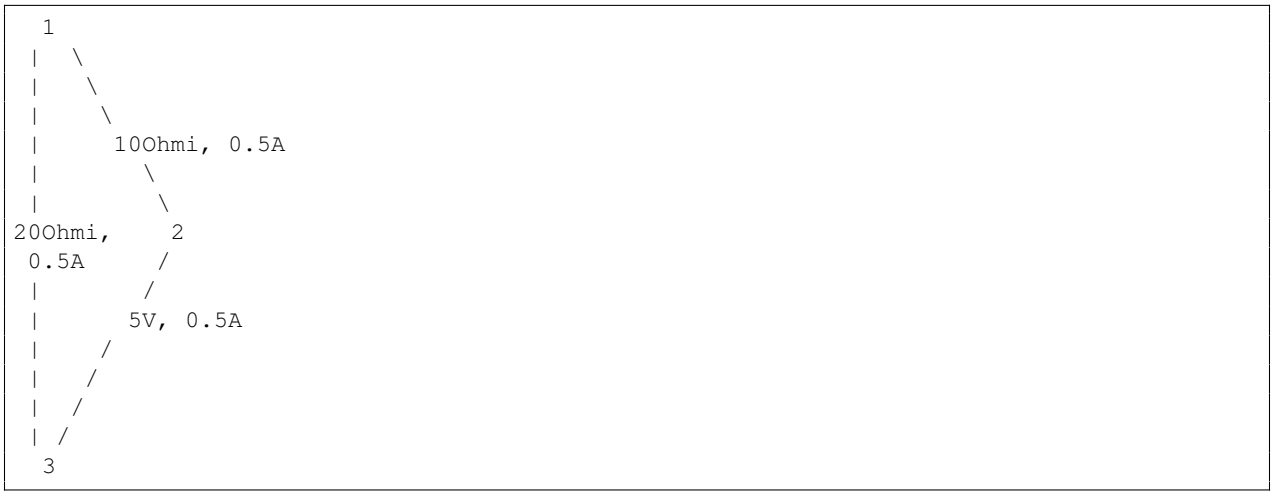
Cum direcția curentului pe firul ce leagă nodurile 1 și 2 este de 0.5A și vom avea o rezistență de 10Ω, putem calcula căderea de tensiune pe aceasta ca fiind $0.5A * 10\Omega = 5V$.

Pentru firul ce leagă nodurile 2 și 3 avem o sursă de tensiune de 5V. Căderea de tensiune pe surse este pe direcție inversă față de cea pe consumatori (precum rezistența) deci o vom pune in celălalt membru al ecuației.

Pentru firul ce leagă nodurile 3 și 1 avem un curent de 0.5A și o rezistență de 20Ω, deci o cădere de tensiune egală cu $0.5A * 20\Omega = 10V$.

$$0.5A * 10\Omega + 0.5A * 20\Omega = 5V \tag{5}$$

$$15V = 5V(F) \tag{6}$$



Exemplul 3

Date de intrare

```
I
6
7
1 2 1
2 3 0.2
2 5 0.6
3 4 0.2
4 5 0.2
5 6 1
6 1 1
```

Date de ieșire

```
Legea 1 a lui Kirchhoff nu se respecta pentru egalitatea 1A = 0.8A in nodul 2.
```

Explicație 3

Circuitul este același ca în primul exemplu, dar în nodul 2 intră un curent de 1A din nodul 1 și ies doi curenți, unul de 0.6A către nodul 5 și unul de 0.2A către nodul 3.

Aplicând formula de mai sus, obținem următoarea relație

$$1A = 0.2A + 0.6A \tag{7}$$

$$1A = 0.8A(F) \tag{8}$$

Problema 4 - Seven segment display

Enunț

Gigel este student în anul II la Automatică și a aflat de la cursul de SOC (Structura si Organizarea Calculatoarelor) despre matricele de leduri. El a lucrat la laborator cu un 7 segment display și a decis să simuleze ceva similar în cod, pentru a putea exersa și acasă. După cum spune și numele, un astfel de display conține 7 led-uri (segmente) contorizate alfabetic de la 'a' la 'g' conform reprezentării de mai jos.

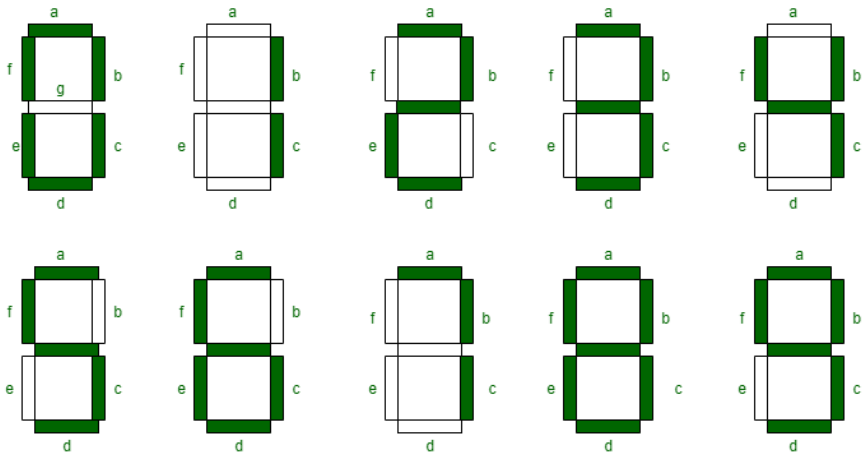


Figure 1: 7SegmentDisplay.
Sursa imaginii: https://www.geeksforgeeks.org/seven-segment-displays

Deoarece segmentele nu sunt pătratice, Gigel dorește să le reprezinte pe mai multe câmpuri în matrice. Pe lângă reprezentarea cifrelor, el dorește să facă și niste translații simple. Cifrele vor fi reprezentate pe o matrice cu n linii și m coloane umplută cu valori de 1 și de 0, în care o valoare de 1 reprezintă un led aprins, iar o valoare de 0 reprezintă un led închis.

Exemplu de reprezentare a segmentelor display-ului

```
0 a a a 0
f 0 0 0 b
f 0 0 0 b
f 0 0 0 b
0 g g g 0
e 0 0 0 c
e 0 0 0 c
e 0 0 0 c
0 d d d 0
```

De asemenea, la afișarea matricii, pentru a fi mai ușor de citit, Gigel vă afișa " " pentru valorile de 0 și "^" pentru valorile de 1.

Un segment de leduri va trebui să conțină atât o lungime cât și o grosime pentru a acoperi matricea. Din input vă va fi oferită lungimea unui segment de leduri, pentru a determina grosimea acestui segment, va trebui să folosiți următoare formulă:

$$Width = [\frac{Len}{3}] \tag{9}$$

Restricții și precizări

- $1 \leq N, M \leq 1000$
- $N = 2 * Len + 3 * Width$
- $M = Len + 2 * Width$
- Numărul de translații începe pe 64 de biți.

Date de intrare

Datele de intrare se vor citi de la **STDIN** altfel:

- N - Numărul de linii din matrice
- M - Numărul de coloane din matrice
- Len - Lungimea unui segment de leduri
- Pe următoarele linii comenzi ce urmează a fi rulate. Șirul de comenzi se termină mereu cu comanda corespunzătoare caracterului 'Q'.

Comenzile posibile pe care programul le poate primi sunt:

- F digit - Va umple matricea cu valorile necesare reprezentării cifrei primite.
- W/A/S/D Count - Va face count translații în sus/stangă/jos/dreapta.
- P - Printează matricea curentă.
- Q - Închide programul.

Date de ieșire

Programul va afișa pentru fiecare comanda de P (Print) la **STDOUT** matricea de leduri stocată curent, conform cerințelor de mai sus. De asemenea, pentru cazurile de eroare vor fi afișate mesaje corespunzătoare:

- "Inputul oferit nu este o cifră.", dacă se oferă ca input comenzii de F (Fill) un input mai mic decât 0 sau mai mare decât 9.
- "Comanda este invalidă.", dacă se inserează o comandă ce nu este menționată mai sus.

Între oricare doi tokeni vizibili din input și din output există exact un spațiu. La finalul oricărei linii din input și din output există un caracter **newline**. La finalul fiecărei cifre afișate, va fi adăugat înca un caracter de tipul **newline**.

Pentru informații suplimentare cu privire la 7 Segment Displays vă recomandăm următoarele site-uri:

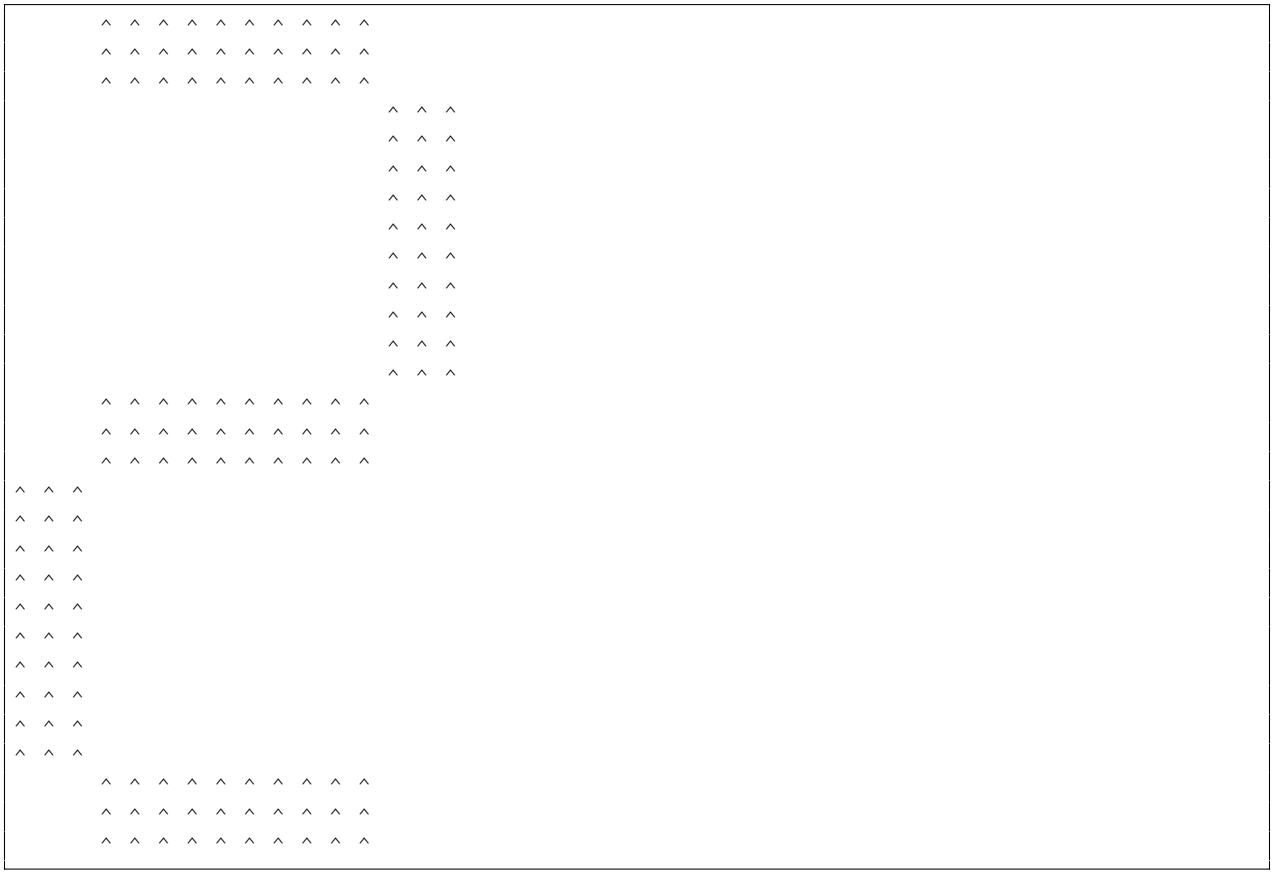
- Wikipedia
- GeeksForGeeks
- Laborator SOC

Exemplul 1

Date de intrare

```
29 16 10
F 2
P
Q
```

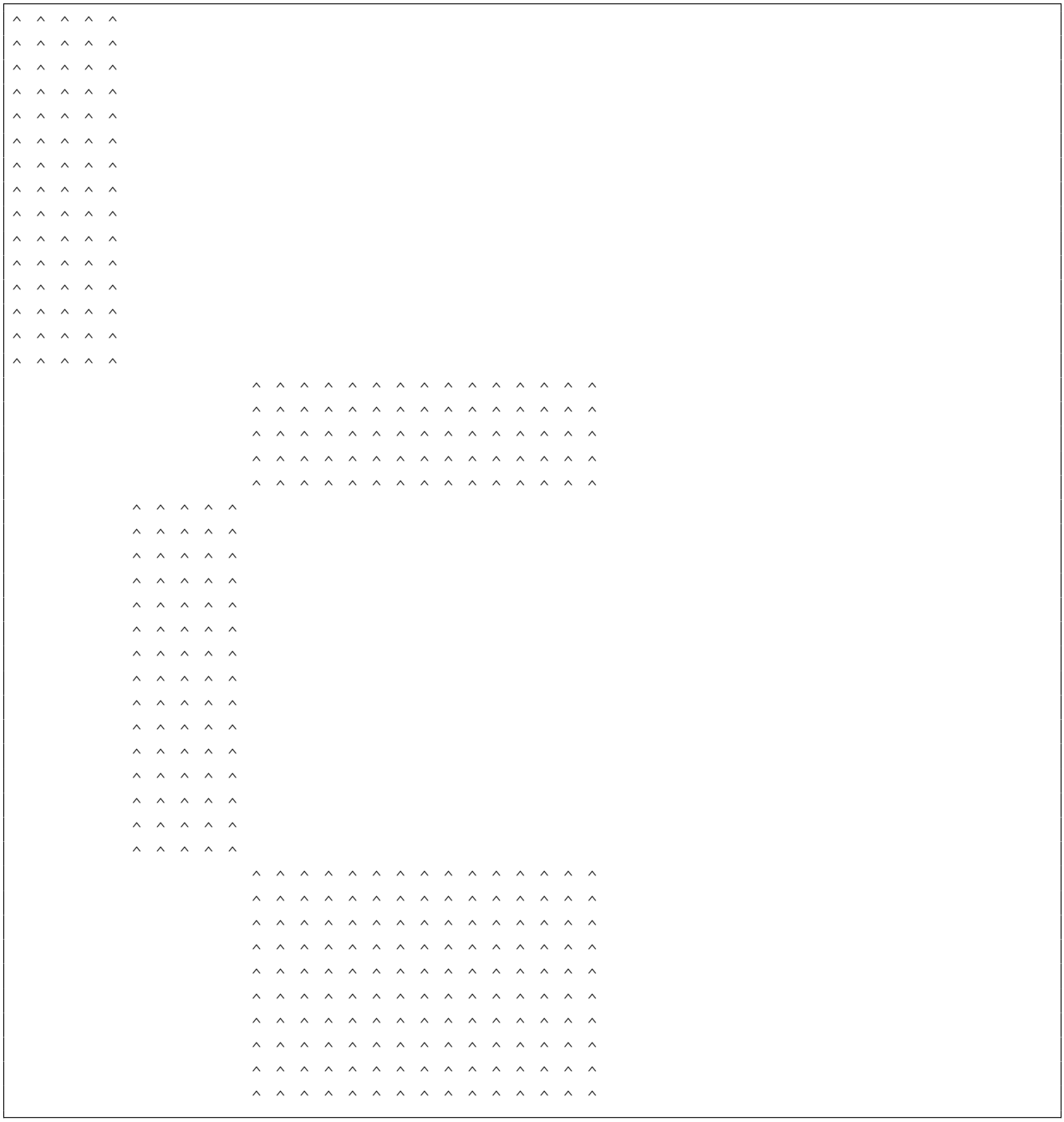
Date de ieșire



Date de intrare

```
45 25 15
F 2
D 5
W 5
P
Q
```

Date de ieșire



Regulament

Regulamentul general al temelor se găsește pe ocw (Temele de casă). Vă rugăm să îl citiți integral înainte de continua cu regulile specifice acestei teme.

Arhivă

Soluția temei se va trimite ca o arhivă **zip**. Numele arhivei trebuie să fie de forma **Grupă_NumePrenume_TemaX.zip** - exemplu: `311CA_NichitaRadu_Tema1.zip`.

Arhiva trebuie să conţină în directorul **RĂDĂCINĂ** doar următoarele:

- Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
  - Regula **build** va compila codul vostru și va genera următoarele executabile:
    - **jocuri_societate** pentru problema 1
    - **forme** pentru problema 2
    - **circuits** pentru problema 3
    - **segment_display** pentru problema 4
  - Regula **clean** va șterge **toate** fișierele generate la build (executabile, binare intermediare etc).
- Un fișier **README** care să conțină prezentarea implementării alese de voi. **NU** copiați bucăți din enunț.

Arhiva temei **NU** va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.

Numele şi extensiile fişierelor trimise **NU** trebuie să conţină spaţii sau majuscule, cu exceptia fișierului README (care are numele scris cu majuscule și nu are extensie).

Nerespectarea oricărei reguli din secțiunea **Arhivă** aduce un punctaj **NUL** (0) pe temă.

Checker

Pentru corectarea aceste teme vom folosi scriptul **check** din arhiva **check_capybara_remastered.zip** din secțiunea de resurse asociată temei. Vă rugăm să citiți **README.md** pentru a știi cum să instalați și utilizați checkerul.

Punctaj

Distribuirea punctajului:

- Problema 1: 15p
- Problema 2: 20p
- Problema 3: 20p
- Problema 4: 25p
- Claritatea și calitatea codului: 10p
- Claritatea explicaţiilor din README: 10p
- Modularizare + implementări deosebite : 10p (punctaj bonus, acordat la corectarea manuală)

ATENȚIE! Punctajul maxim pe temă este **100p**. Acesta reprezintă 0.5p din nota finală la această materie. La această temă se pot obține până la la **110p** (există un bonus de până la 10p acordat pe baza modularizării și a unor implementări deosebite, ce va fi acordat la corectarea manuală). Punctajul bonus de la teme se acordă doar la nota finală, dacă toate condițiile de promovare au fost deja satisfăcute.

Reguli și precizări

- Punctajul pe teste este cel acordat de script-ul **check**, rulat pe **Moodle**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare, etc).
- Punctajul pe calitatea explicațiilor și a codului se acordă în mai multe etape:
  - **corectare automată**
    - Checkerul va încerca să detecteze în mod automat probleme legate de coding style și alte aspecte de organizare a codului.
    - Acesta va puncta cu maxim 20p dacă nu sunt probleme detectate în mod automat.
    - Punctajul se va acorda proporțional cu numărul de puncte acumulate pe teste din cele 80p.
    - Checkerul poate să aplice însă și penalizări (exemplu pentru warninguri la compilare) sau alte probleme descoperite la runtime.
  - **corectare manuală**
    - Tema va fi corectată manual și se vor verifica și alte aspecte pe care checkerul nu le poate prinde. Recomandăm să parcurgeți cu atenție tutorialul de coding-style de pe ocw.cs.pub.ro.
    - Codul sursă trebuie să fie însoțit de un fișier README care trebuie să conțină informațiile utile pentru înțelegerea funcționalității, modului de implementare și utilizare a programului. Acesta evaluează, de asemenea, abilitatea voastră de a documenta complet și concis programele pe care le produceți și va fi evaluat, in mod analog CS, de către echipa de asistenți. In funcție de calitatea documentației, se vor aplica depunctări sau bonusuri.
    - La corectarea manuală se va acorda un bonus de maximum 10 puncte pentru modularizare. Deprinderea de a scrie cod sursă de calitate, este un obiectiv important al materiei. Sursele greu de înteles, modularizate neadecvat sau care prezintă hardcodări care pot afecta semnificativ mentenabilitatea programului cerut, pot fi depunctate adițional. Penalizarea pentru modularizare defectuoasă sau absentă complet, poate să fie oricât de mare.
    - În această etapă se pot aplica depunctări oricât de mari (chiar și totale). Orice rezultat/punctaj acordat automat de checker, poate fi anulat sau suprascris de către echipa de PCLP la corectarea manuală.
    - O temă care **NU** compilează cu -Wall -Wextra este depunctată la corectarea manuală cu 5p (punctajul echivalent pentru warnings).
    - **Tema trebuie să reprezinte exclusiv munca studentului!** Echipa va aplica regulamentele în vigoare pentru situațiile de fraudă/plagiat, pentru orice nerespectare a acestei reguli, incluzând, dar nelimitându-se la: copierea de la colegi (se aplică pentru ambele persoane implicate) sau din alte surse a unor părți din temă, prezentarea ca rezultat personal a oricărei bucăți de cod provenită din alte surse necitate sau neaprobate în prealabil (site-uri web, alte persoane, cod generat folosit AI/LLM-uri etc.).

Alte precizări

- Implementarea se va face in limbajul C, iar tema va fi compilată şi testată **DOAR** într-un mediu **LINUX**. Nerespectarea acestor reguli aduce un punctaj **NUL**.
- Tema trebuie trimisă sub forma unei arhive pe site-ul cursului curs.upb.ro.
- Tema poate fi submisă de oricâte ori fără depunctări până la deadline. Mai multe detalii se găsesc în regulamentul de pe ocw.
- O temă care **NU** compilează **NU** va fi punctată.
- O temă care compilează, dar care **NU** trece niciun test **NU** va fi punctată.
- Punctajul pe teste este cel acordat de **check** rulat pe **platforma remote a echipei de PCLP**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).
- Ultima temă submisă pe Moodle poate fi rulată de către responsabili/echipa de PCLP de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. În cazul obținerii punctajelor diferite la rulări multiple, vom păstra minimul dintre punctaje. Vă recomandăm să verificați local tema de mai multe ori pentru a verifica că punctajul este mereu același, apoi să încărcați tema. De asemenea,

verificați că punctajul de pe platformă este cel așteptat - singurul punctaj / feedback relevant este cel de pe platforma remote PCLP.