Poli TicTacToe

Responsabili:

- George Popescu [mailto:george.apopescu97@gmail.com]
- Darius Neațu [mailto:darius.neatu@cti.pub.ro]

Deadline: 26.11.2018 23:55

Actualizări

- **06.11.2018 23:35** - Actualizare arhivă check_gigel.zip [https://ocw.cs.pub.ro/courses/_media/programare/teme_2018/check_gigel.zip]. Vă rugăm să descarcați din nou arhiva. În unele fișiere caracterul apostrof(') era înlocuit cu altceva.
- **06.11.2018 22:33** - Adăugare precizări suplimentare la task 2 despre modul de câștigare a macroboard-ului.
- **06.11.2018 00:00** - **NU** există actualizări.

Obiective

- Exersarea lucrului cu tablouri și funcții.
- Implementarea unor algoritmi conform unor specificații, dar și găsirea unor soluții pentru anumite cerințe.
- Simularea unui joc de tipul X și 0 (modificat) printr-un program scris în C. Tratarea cazurilor (de eroare) multiple care pot apărea într-o aplicație complexă.
- Exersarea aptitudinilor de a scrie cod clar, modularizat. Think twice before you jump to the code!

CITEȘTE!

Tema **NU** interzice folosirea celorlalte concepte din C care nu sunt menționate mai sus.

Aveți voie să folosiți orice, atâta timp cât respectați convențiile stabilitate la curs/laborator (ex. nu aveți voie cu variabile globale sau goto). Puteți folosi macrouri, pointeri/pointeri la funcții, alocare dinamică, stringuri, structuri, funcții variadice, funcții polimorfice etc dacă vă ajută și le utilizați într-un mod **corespunzător**. Folosirea greșită se depunctează (ex. alocare dinamică fără eliberare de memorie).

Menționăm că pentru testare (pe vmchecker) se folosește o mașină virtuală pe 32 de biți. În caz că sistemul vostru de operare de pe mașina fizică este pe 64 de biți, sugerăm să faceți testarea finală și pe o mașină (virtuală sau nu) de 32 de biți.

Compilați cu flag-urile din exemplul următor:

```
gcc -Wall -Wextra -std=c99 main.c -o gigel
```

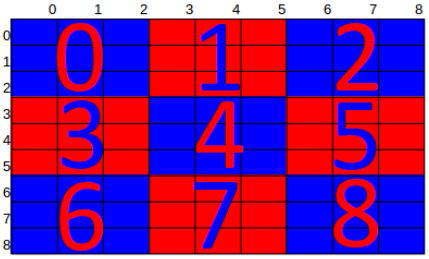
- **-Wall -Wextra** : Vă arată warning-uri pentru a putea primi indicații suplimentare despre codul vostru (ex. posibile probleme). Va trebui să modificați codul a.i. să nu mai aveți warning-uri.
- **-std=c99** : Din păcate versiunea de compilator de pe vmchecker este foarte veche si are standardul c89 default. Noi folosim c99, de aceea **trebuie** să adăugați acest flag la compilare.

Sugerăm să citiți cu atenție **TOATĂ** secțiunea Testare [https://ocw.cs.pub.ro/courses/programare/teme_2018/tema2_2018_ca#testare], înainte de a folosi checkerul local.

Enunț

Gigel și Maria vă propun să jucați o variantă nouă a jocului X și O, care are următoarele reguli: Tabla de joc are dimensiune n^2 x n^2 și se numește **board**. Acesta este împărțit în n x n table mai mici numite **miniboard**-uri, fiecare având fix n x n elemente (celule). În fiecare miniboard se va desfășura un joc de X și O.

Exemplu: Pentru n = 3, un board are dimensiune 3^2 x 3^2 = 9 x 9, iar un miniboard are dimensiune 3 x 3. Acest exemplu este ilustrat în figura de mai jos, unde cele 9 miniboard-uri au fost numerotate de la 0 la 8 (având culori diferite pentru a scoate în evidență celule componente).

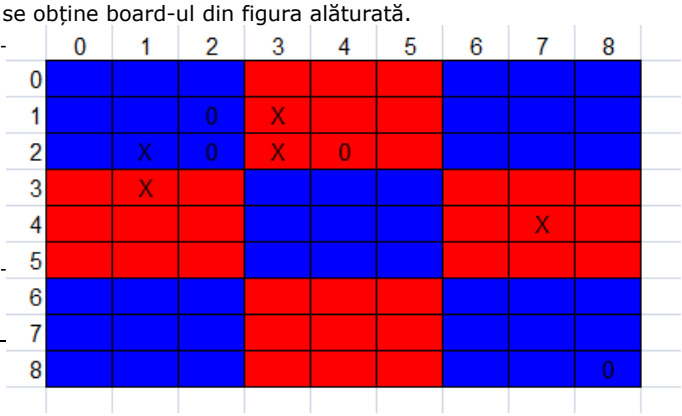


Pe acest board se efectuează m mutări de forma: <nume> x y

O astfel de mutare înseamnă că jucătorul cu numele dat alege să seteze elementul de pe poziția (x, y) din board cu X sau 0 (X pentru primul jucător, 0 pentru cel de-al doilea jucător). Ca în orice joc de X și 0, jucătorii trebuie să mute **alternativ**.

Exemplu - n = 3, m = 9. Dacă șirul de mutări este următorul, atunci se obține board-ul din figura alăturată.

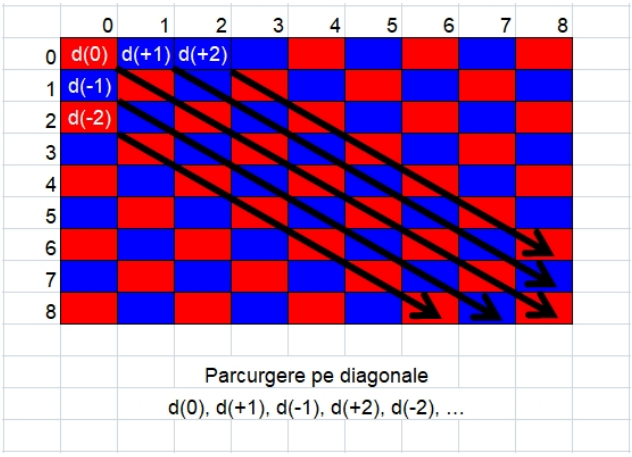
```
3 9
X 3 1
0 1 2
X 2 3
0 2 4
X 1 3
0 2 2
X 2 1
0 8 8
X 4 7
```



Task 1 (de la Gigel)

Primul vostru task este să validați aceste mutări verificând și tratând următoarele reguli/cazuri:

- **Inițial** board-ul este **liber**, deci se poate muta în orice celulă.
- O mutare validă înseamnă să folosească o celulă liberă din board.
- În acest joc începe **mereu** jucătorul care pune X în board.
- Jucătorii trebuie **să mute alternativ**.
  - **Dacă** mutarea este **validă**, atunci această se va **executa** (se va pune X sau 0 în board) și va urma rândul celuilalt jucător.
  - **Dacă** jucătorul care este la mutare propune **o mutare invalidă**, atunci se va trata special acest caz astfel:
    - 1. Se va afișa **un mesaj de eroare** astfel:
      - 1.1 Dacă cel puțin unul dintre x și y nu reprezintă indici valizi pentru board, atunci se va afișa pe ecran mesajul: "INVALID INDEX" (fără ghilimele).
      - 1.2 Dacă x și y reprezintă indecși valizi, dar celula de coordonate (x, y) din board este deja ocupată, atunci se va afișa mesajul: "NOT AN EMPTY CELL" (fără ghilimele).
    - 2. Jucătorul își pierde dreptul de a spune altă mutare în această rundă, așa că i se va atribui **automat** o mutare obținută după următorul algoritm de tip round-robin:
      - Se parcurge matricea pe **diagonale**, iar la găsirea **primului element liber** ne oprim. Celula găsită de coodonate (p, q) va fi considerată mutare pentru jucătorul curent și va fi efectuată.
      - Parcurgerea pe diagonale se face face mereu de **sus** în **jos** și de la **stânga** la **dreapta** (procedeu ilustrat în figura alăturată). Vom considera doar diagonalele paralele cu cea principală în următoarea ordine:
        - 1. Se parcurge mai întâi diagonala principală (d_0)
        - 2. Se parcurg cele 2 diagonale adiacente (la distantă 1), mai întâi cea de deasupra diagonalei principale, apoi cea de dedesubpt. (d_1, d_{-1})
        - 3. Se parcurg umătoarele 2 diagonale (la distantă 2), mai întâi cea de deasupra diagonalei principale, apoi cea de dedesubpt. (d_2, d_{-2})
        - Continuăm procedeul până găsim un element liber sau până epuizăm **toate** diagonalele (am parcurs complet matricea).
          - În caz că am găsit o celulă liberă, se va folosi după cum a fost descris anterior.
          - În caz că **nu** mai există celule libere în matrice, se vor face următoarele 2 lucruri:
            - 1. Se va afișa pe ecran mesajul: "FULL BOARD" (fără ghilimele).
            - 2. Următoarele mutări rămase vor fi citite și **ignorate** - nu se vor executa. (Deoarece jocul s-a terminat).



  - Dacă mutarea curentă citită nu corespunde cu jucătorul care este la mutare, atunci:
    - 1. Se va afișa pe ecran mesajul: "NOT YOUR TURN" (fără ghilimele).
    - 2. Această mutare **nu** se va executa și se va citi următoarea mutare.
- Jocul se termină când s-au citit toate mutările.

Rezultatul produs de task 1 este reprezentat de mesajele de eroare afișate de voi.

Fie un joc cu n = 3. Consirăm 8 mutări.

input	output

3 8 X 0 0 0 0 3 X 0 1 0 0 4 X 0 2 0 0 5 0 0 0 X 0 0	NOT YOUR TURN NOT AN EMPTY CELL

Explicație :

- Executând pas cu pas, observăm că:
  - mutarea **0 0 0** este invalidă și produce mesajul **NOT YOUR TURN**
  - mutarea **X 0 0** este invalidă și produce mesajul **NOT AN EMPTY CELL**
    - Se găsește primul element liber parcurgând pe diagonale: $(move_x, move_y) => (1, 1)$
- Toate celelalte mutări au fost valide și au fost executate așa cum au fost date.

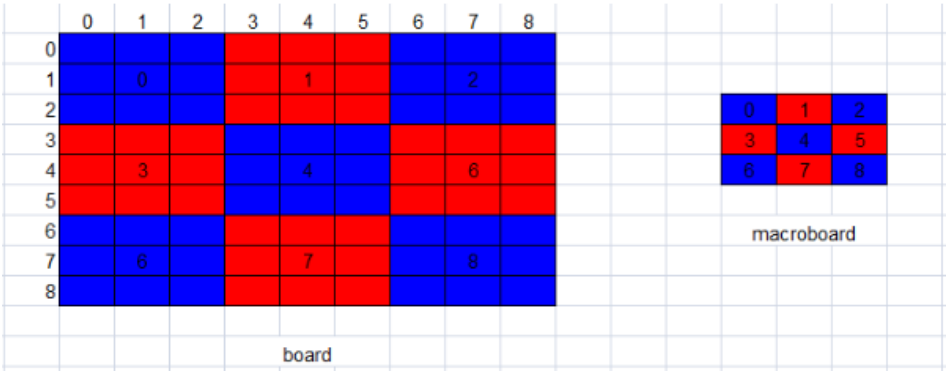
Configurația finală a board-ului este următoarea:

```
  012 345 678
0 XXX 000 ---
1 -X- --- ---
2 --- --- ---

3 --- --- ---
4 --- --- ---
5 --- --- ---

6 --- --- ---
7 --- --- ---
8 --- --- ---
```

Task 2 (de la Maria)

După ce am efectuat cele m mutări de la task 1, Maria dorește să vedem care jucător a câștigat. Pentru aceasta vom introduce un tablou de dimensiune n x n, pe care îl vom numi **macroboard**. Fiecare **miniboard** va avea **o unică celulă** asociată în macroboard.

Exemplu - pentru n = 3, asocierea este ilustrată în figura următoare.



După cum am spus anterior, în fiecare miniboard se va desfășura un joc de X și 0 de dimensiune n x n. Într-un miniboard câștigă jucătorul care are cel puțin **o linie**, **o coloană** sau **o diagonală** (cea principală sau cea secundară), plină cu X (pentru jucătorul cu X) sau cu 0 (pentru jucătorul cu 0).

Menționăm că un miniboard este câștigat de **primul** jucător care face o linie, o coloană sau o diagonală în acest miniboard. Se poate pune în continuare mutări într-un miniboard câștigat, dar acesta rămâne în continuare a celui care l-a câștigat inițial.

Pentru un miniboard câștigat, se va pune X sau 0 în macroaboard, corespunzător jucătorului care **a câștigat**.

În caz de remiză într-un miniboard, în macroboard se va pune caracterul **-** (liniuță). Considerăm remiză orice caz în care **nu** există **un** câștigător.

După completarea macroboard-ului, care are tot dimensiune n x n, aplicăm următoarea regulă: jocul este câștigat de jucătorul care are cel puțin o linie, o coloană sau o diagonală (cea principală sau secundară) plină cu caracterul său în macroboard la **finalul jocului**.

Menționăm că jocul/macroboard-ul **poate** fi câștigat de **un singur** jucător. Acesta se stabilește la **finalul** execuției celor m mutări!

Pentru a rezolva task 2 se cer următoarele:

- 1. Să se afișeze **configurația finală a macroboard-ului** după executarea celor m mutări de la task 1.
- 2. Să se determine **cine a câștigat** și să se afișeze pe ecran **unul** dintre mesajele
  - "X won"
  - "0 won"
  - "Draw again! Let's play darts!"

Fie un joc cu n = 3. Consirăm 8 mutări.

input	output
3 8 X 0 0 0 0 3 X 0 1 0 0 4 X 0 2 0 0 5 0 0 0 X 0 0	NOT YOUR TURN NOT AN EMPTY CELL X0- ---'--- Draw again! Let's play darts!

Explicație :

- Executând pas cu pas, observăm că:
  - mutarea **0 0 0** este invalidă și produce mesajul **NOT YOUR TURN**
  - mutarea **X 0 0** este invalidă și produce mesajul **NOT AN EMPTY CELL**
    - Se găsește primul element liber parcurgând pe diagonale: $(move_x, move_y) => (1, 1)$
- Toate celelalte mutări au fost valide și au fost executate așa cum au fost date.
- Configurația finală pentru BOARD și MACROBOARD se poate vedea mai jos.
  - X a câștigat miniboard-ul 0, iar 0 a câștigat miniboard-ul 1.
  - Întrucât nu există **un câștigător** în macroboard, jocul s-a terminat remiză.

```
  012 345 678
0 XXX 000 ---
1 -X- --- ---
2 --- --- ---

3 --- --- ---
4 --- --- ---
5 --- --- ---

6 --- --- ---
7 --- --- ---
8 --- --- ---

     BOARD
```

```
   012
0 X0-
1 ---
2 ---

 MACROBOARD
```

Task 3 (tot de la Gigel)

Deoarece Gigel poate folosi acum rezolvarea voastră să verifice șirul de mutări pe care le face cu Maria și poate determina câștigătorul, el vrea să vă mai dea încă un task **greu**.

Gigel se întreabă cât de inteligenți au fost ei doi când au generat acest șir de mutări. Probabil că ar fi foarte greu să calculați asta, așa că el dorește să calculați pentru jucătorul X și jucătorul 0 câte un coeficient de atenție, care se definește astfel: **attention** = numărul de situații în care player a câștigat prin alegerea **sa** un miniboard raportat la numărul de runde jucate de el.

Pentru a rezolva task 3 se cere să se afișeze 2 linii:

```
X <attention>
0 <attention>
```

Dacă nu se poate calcula coeficientul attention pentru un jucător, atunci se va afișa o linie de forma:

```
<jucator> N/A
```

- Deoarece X începe jocul mereu, se va afișa mai întâi linia corespunzătoare lui X. Dacă se poate calcula coeficientul de atenție, se va afișă cu **10 zecimale exacte**.

Fie un joc cu n = 3. Consirăm 8 mutări.

input	output
3 8 X 0 0 0 0 3 X 0 1 0 0 4 X 0 2 0 0 5 0 0 0 X 0 0	NOT YOUR TURN NOT AN EMPTY CELL X0- ---'--- Draw again! Let's play darts! X 0.2500000000 0 0.3333333333

Explicație :

- Executând pas cu pas, observăm că:
  - mutarea **0 0 0** este invalidă și produce mesajul **NOT YOUR TURN**

- mutarea **X 0 0** este invalidă și produce mesajul **NOT AN EMPTY CELL**
  - Se găsește primul element liber parcurgând pe diagonale: $(move_x, move_y) => (1, 1)$

- Toate celelalte mutări au fost valide și au fost executate așa cum au fost date.
- Configurația finală pentru BOARD și MACROBOARD se poate vedea mai jos.
  - X a câștigat miniboard-ul 0, iar 0 a câștigat miniboard-ul 1.
  - Întrucât nu există **un câștigător** în macroboard, jocul s-a terminat remiză.
- Fiecare a câștigat câte un miniboard; X a jucat 4 runde, 0 a jucat 3 runde.
  - $1/4 = 0.2500000000$
  - $1/3 = 0.3333333333$

```
  012 345 678
0 XXX 000 ---
1 -X- --- ---
2 --- --- ---

3 --- --- ---
4 --- --- ---
5 --- --- ---

6 --- --- ---
7 --- --- ---
8 --- --- ---

     BOARD
```

```
   012
0 X0-
1 ---
2 ---

MACROBOARD
```

Cerință

Se citesc de la tastatură numerele **n**,**m**, apoi cele **m mutări**. Se cere să se aplice cele m mutări, să se rezolve cele m mutări și să se afișeze în ordine răspunsurile pentru cele 3 task-uri:

- **Primele q linii** din fișier vor conține **mesajele de eroare** afișate la task 1.
- Următoarele $n + 1$ linii vor conține răspunsul pentru task 2
  - Primele n linii conțin **macroboard-ul** (câte n elemente pe o linie).
  - Următoarea linie conține mesajul prin care se anunță **cine a câștigat**.
- Fiecare dintre următoarele **2 linii** conține numele unui jucător și coeficientul de atenție.

Format date de intrare

```
n m
player_1 x_1 y_1
player_2 x_2 y_2
...
player_m x_m y_m
```

Prima linie conține numerele n și m despărțite prin câte un spațiu.

Următoarele m linii respectă formatul: un caracter (X sau 0), un spațiu, un număr întreg (x), un spațiu, un număr întreg (y).

Format date de ieșire

```
<error_message_1>
<error_message_2>
....
<error_message_q>
macroboard[0][0]macroboard[0][1]...macroboard[0][n-1]
macroboard[1][0]macroboard[1][1]...macroboard[1][n-1]
..........
macroboard[n-1][0]macroboard[n-1][1]...macroboard[n-1][n-1]
<win_message or draw message>
X <attention_x or N/A>
0 <attention_0 or N/A>
```

Să presupunem că la task 1 se vor detecta q erori, pentru fiecare afișându-se un mesaj. Se va afișa fiecare mesaj pe o linie! (nici un caracter în plus)

Urmează n linii, fiecare conținând câte n caractere. Aceste n^2 elemente reprezintă macroboard-ul.

Următoarea linie conține mesajul prin care se anunță victori sau remiza din jocul curent.

Următoarele 2 linii conțin fiecare: un caracter (X sau O), un spațiu, apoi fie coeficientul de atenție cu fix 10 zecimale, fie mesajul "N/A".

Restricții și precizări

- $2 < n <= 10$
- $1 < m <= 10^6$
- pentru orice mutare de forma $player_i x_i, y_i$, avem că
  - $player_i \in \{'X', '0'\}$
  - x_i, y_i sunt numere întregi pe 32 de biți
- **Toate** datele se vor citi de la **tastatură**. **Toate** rezultatele se vor afișa pe **ecran**. Checker-ul va face redirectările necesare astfel încât datele să fie citite din fișiere (*.in) și să fie afișate în fișier (*.out), apoi se va compara cu rezultatele din fișierele de referință (*.ref).
- **Sugerăm** să faceți pe foaie exemplele și testele mici pentru a vă asigura că ați înteles **foarte bine** enunțul înainte să vă apucați de scris cod.

Exemple

Mai jos se află o listă cu câteva exemple explicate pe scurt.

Toate aceste exemple se găsesc în arhiva de testare check_gigel.zip.

Pentru fiecare exemplu se găsesc:

- fișierul de intrare (**xy-example.in**)
- fișierul de referință (**xy-example.ref**)
- un fișier care conține explicații **sumare** despre ce se îmtâmplă la fiecare pas (**xy-example.summary**)
- un fișier care conține, în plus față de summary, stare board-ului la fiecare pas (**xy-example.full**)

Fie un joc cu n = 3. Consirăm 8 mutări.

00-example.in	00-example.ref
3 8 X 0 0 0 0 3 X 0 1 0 0 4 X 0 2 0 0 5 0 0 0 X 0 0	NOT YOUR TURN NOT AN EMPTY CELL X0- ---'--- Draw again! Let's play darts! X 0.2500000000 0 0.3333333333

Explicație :

- Executând pas cu pas, observăm că:
  - mutarea **0 0 0** este invalidă și produce mesajul **NOT YOUR TURN**
  - mutarea **X 0 0** este invalidă și produce mesajul **NOT AN EMPTY CELL**
    - Se găsește primul element liber parcurgând pe diagonale: $(move_x, move_y) => (1, 1)$
- Toate celelalte mutări au fost valide și au fost executate așa cum au fost date.
- Configurația finală pentru BOARD și MACROBOARD se poate vedea mai jos.
  - X a câștigat miniboard-ul 0, iar 0 a câștigat miniboard-ul 1.
  - Întrucât nu există **un câștigător** în macroboard, jocul s-a terminat remiză.
- Fiecare a câștigat câte un miniboard; X a jucat 4 runde, 0 a jucat 3 runde.
  - $1/4 = 0.2500000000$
  - $1/3 = 0.3333333333$

```
  012 345 678
0 XXX 000 ---
1 -X- --- ---
2 --- --- ---

3 --- --- ---
4 --- --- ---
5 --- --- ---

6 --- --- ---
7 --- --- ---
8 --- --- ---

     BOARD
```

```
    012
  0 X0-
  1 ---
  2 ---

 MACROBOARD
```

Fie un joc cu n = 4. Consirăm 8 mutări.

00-example.in	00-example.ref
4 8 X 0 0 0 1 0 X 0 1 0 1 1 X 0 2 0 1 2 X 0 3 0 1 3	X--- ---- ---- '--- Draw again! Let's play darts! X 0.2500000000 0 0.0000000000

Explicație :

- Executând pas cu pas, observăm că toate mutările sun valide!
- Configurația finală pentru BOARD și MACROBOARD se poate vedea mai jos. ATENȚIE! Pentru aliniere in exemplu, coloanele 10, 11, 12, 13, 14, 15 au fost notate cu ultima cifră (0, 1, 2, 3, 4, 5).
  - X a câștigat miniboard-ul 0.
  - Întrucât nu există **un câștigător** în macroboard, jocul s-a terminat remiză.
- X a jucat 4 runde (și a câștigat un miniboard), 0 a jucat 4 runde.
  - $1/4 = 0.2500000000$
  - $0/4 = 0.0000000000$

```
   0123 4567 8901 2345
 0 XXXX ---- ---- ----
 1 0000 ---- ---- ----
 2 ---- ---- ---- ----
 3 ---- ---- ---- ----

 4 ---- ---- ---- ----
 5 ---- ---- ---- ----
 6 ---- ---- ---- ----
 7 ---- ---- ---- ----

 8 ---- ---- ---- ----
 9 ---- ---- ---- ----
10 ---- ---- ---- ----
11 ---- ---- ---- ----

12 ---- ---- ---- ----
13 ---- ---- ---- ----
14 ---- ---- ---- ----
15 ---- ---- ---- ----

     BOARD
```

```
  0123
0 X---
1 ----
2 ----
3 ----

 MACROBOARD
```

Testare

Testarea temei se va face folosind un script de evaluare automată, ce poate fi găsit în check_gigel.zip [https://ocw.cs.pub.ro/courses/_media/programare/teme_2018/check_gigel.zip].

Compilați cu flag-urile din exemplul următor:

```
gcc -Wall -Wextra -std=c99 main.c -o gigel
```

- **-Wall -Wextra** : Vă arată warning-uri pentru a putea primi indicații suplimentare despre codul vostru (ex. posibile probleme). Va trebui să modificați codul a.i. să nu mai aveți warning-uri.
- **-std=c99** : Din păcate versiunea de compilator de pe vmchecker este foarte veche si are standardul c89 default. Noi folosim c99, de aceea **trebuie** să adăugați acest flag la compilare.

Instrucţiuni de utilizare

- Arhiva se dezarhivează în directorul vostru de lucru (acolo unde este compilat executabilul `gigel`).
- În acest director **trebuie** să aveți conținutul temei voastre: surse, fișier Makefile și README.
- Arhiva conţine un fişier `check.sh`, din care se porneşte operaţia de testare (`./check.sh`).

Menționăm că pentru testare (pe vmchecker) se folosește o mașină virtuală pe 32 de biți. În caz că sistemul vostru de operare de pe mașina fizică este pe 64 de biți, sugerăm să faceți testarea finală și pe o mașină (virtuală sau nu) de 32 de biți.

Exemplu de utilizare:

```
darius@pc ~ $ ./check.sh
```

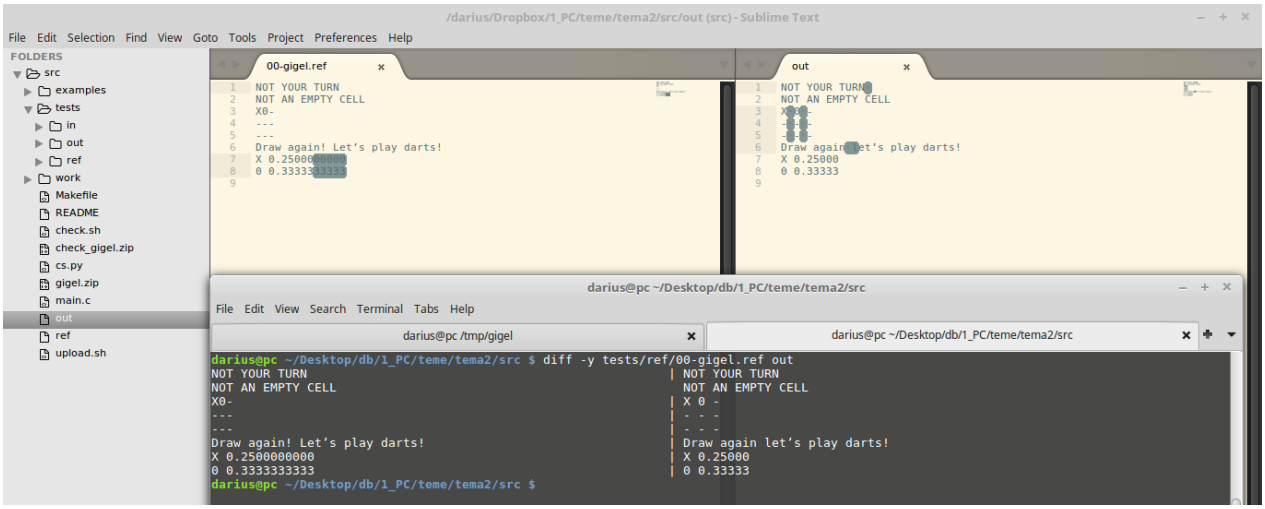
- Arhiva mai conține un script numit `cs.py`, prezentat în pagina de Coding Style de pe ocw.
- Dacă apar erori și testarea eșuează, puteți să vă uitați în directorul `tests` și să comparați rezultatele voastre (fișierele `tests/out/*.out`), cu cele ale implementării de referință (fișierele `tests/ref/*.ref`).

ea să-l rulați separat și să experimentați cu el și alte situații.

Dacă doriți să testați manual folosind redirectări (fără a introduce de fiecare dată datele de la tastatură), puteți folosi această metodă.

```
darius@pc ~ $ make build # imi va creea executabilul gigel

darius@pc ~ $ ./gigel < tests/in/00-gigel.in # voi rula pe gigel cu inputul din fisierul tests/in/00-gigel.in
                                             # rezultatul se va afisa pe ecran

darius@pc ~ $ ./gigel < tests/in/00-gigel.in > out # voi rula pe gigel cu inputul din fisierul tests/in/00-gigel.in
                                                   # rezultatul va fi salvat in fisierul out (pe care pot da cat sau il pot deschide in Sublie)

darius@pc ~ $ diff tests/ref/00-gigel.ref out # diff compara linie cu linie cele 2 fisiere
                                              # daca nu afiseaza ceva (erori) atunci sunt indentice
                                              # altfel arata cum este linia in primul fisier si cum arata in al 2lea
                                              # DACA fisierele au linii mici, se pot compara in paralel (pe coloane), folosind parametrul -y.
```

Un exemplu de comparare se află in poza următoare.



Barem corectare

- Criteriile de notare sunt următoarele:
  - Teste: 95p
    - Punctajul pe un test se acordă **dacă și numai dacă** au fost rezolvate corect **toate** cele 3 task-uri.
  - README: 5p
    - Punctajul pe README se acordă pentru conținut. **NU** trebuie să exagerați cu detaliile, **NU** trebuie să copiați enunțul. Spuneți ce alegeri ați făcut și care e ideea voastră de rezolvare pentru fiecare task. Care au fost dificultățile, care au fost soluțiile pe care le-ați găsit și (dacă sunt mai multe) de ce ați ales-o pe cea curentă.

Se pot aplica depunctări suplimentare pentru orice nu respectă standardele stabilite la curs/ laborator (ex. 5 puncte pentru warning-uri de compilare).

Punctajul pe teste este cel acordat pe vmchecker. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).

Punctajul pe README va fi stabilit în urma corectării manuale. Checker-ul doar vă reamintește dacă nu aveți README sau aveți un README gol în arhivă, pentru a submite unul în arhiva finală.

Checker-ul va încerca să detecteze în mod automat cele mai frecvente greșeli de coding style. Dacă checkerul penalizează cu **15p** sursa voastră pentru coding style, atunci această decizie este **finală** . În caz contrar, checkerul nu va scădea puncte pentru coding style, dar la corectarea manuală se pot depuncta pentru lucruri suplimentare precum:

- Lucrurile stabilite la curs/laborator (ex. variabile globale, goto).
- Alte aspecte legat de coding style (ex. nume nesugestive pentru pentru funcții si variabile - ex. nume care NU sunt sugestie: var1, var2, f1, f2).
- Se pot aplica penalizări de până la 7p pentru lipsa comentariilor din cod sau pentru comentarii irelevante (ex. nu spuneți că funcția f primește ca parametru un int, un char și un double - se vede pe cod). Câteva comentarii per funcție sunt necesare pentru a descrie funcționalitatea acesteia (daca este mică). În cazuri execptionale, dacă o funcție face lucruri foarte complicate, atunci se pot adăuga comentarii în plus.
- Pentru aceste aspecte, vă punem la dispoziție un model de rezolvare de laborator, redactat ca și o temă (homework_template.zip [https://ocw.cs.pub.ro/courses/_media/programare/teme_2018/homework_template.zip]). Atenție! Este

doar un model dintre multe altele posibile, **NU trebuie** să faceți întocmai ca în acest model, dar recomandăm să îl incercați dacă doriți să lucrați foarte mult la aspectul de redactare al codului.
- Se pot aplica depunctări care nu apar în lista de mai sus!

Regulament

Copierea parţială sau totală a unei rezolvări din altă sursă va atrage după sine anularea punctajelor pentru **toate temele de casă** , atât pentru cel care a copiat, cât şi pentru sursa acestuia.

- Regulamentul general se găsește aici [https://ocw.cs.pub.ro/courses/programare/regulament-ca].
- Tema se va implementa DOAR în limbajul **C**. Va fi compilat și testat DOAR într-un mediu **LINUX**. Nerespectarea acestor reguli aduce un punctaj NUL.
- Temele vor fi trimise OBLIGATORIU pe vmchecker **ȘI** pe site-ul de curs (moodle), in sectiunea dedicata temei respective.
- Fișierele temei trebuie OBLIGATORIU împachetate într-o arhiva de tip '.zip', cu numele **Grupa_NumePrenume_Tema2.zip** (exemplu: **369CA_PopescuGigel_Tema2.zip**).
- Arhiva va trebui să conţină **in directorul RADACINA** doar urmatoarele:
  - 1. Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
  - 2. Un fișier `Makefile` care să conțină regulile `build` și `clean`. Regula `build` va compila programul într-un executabil cu numele **`gigel`**. Regula `clean` va șterge executabilul și eventual toate binarele intermediare (fișiere obiect) generate de voi. Un exemplu concret de Makefile se găsește în arhiva homework_template.zip [https://ocw.cs.pub.ro/courses/_media/programare/teme_2018/homework_template.zip].
  - 3. Un fișier `README` care să conțină prezentarea implementării alese de voi. NU copiați bucăți din enunț în README.
  - Nerespectarea regulilor 1 și 2 aduce un punctaj NUL pe temă.
- Arhiva temei NU va conține fișiere binare. Nerespectarea acestei reguli aduce un punctaj NUL.
- O temă care NU compilează pe vmchecker [https://elf.cs.pub.ro/vmchecker/] NU va fi punctată.
- O temă care compilează, dar care NU trece niciun test pe vmchecker [https://elf.cs.pub.ro/vmchecker/], NU va fi punctată. Punctele pe teste sunt cele de pe vmchecker [https://elf.cs.pub.ro/vmchecker/]. Ultima temă submisă pe vmchecker poate fi rulată de către responsabili de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. Vă recomandăm să verificați **local** tema de mai multe ori pentru a verifica că punctajul este mereu același.

programare/teme_2018/tema2_2018_ca.txt · Last modified: 2018/11/18 14:56 by darius.neatu