PROGRAMAREA CALCULATOARELOR ȘI LIMBAJE DE PROGRAMARE 1

- TEMA 2 -

Profesor coordonator

Carmen ODUBĂȘTEANU

Responsabili

Axenia Alexandru

Florea Daria-Mihaela

Ion Alexandra Ana-Maria

Simescu Andrei

Termen de predare

Vineri 30.12.2022, ora 23:55

2022 - 2023

CUPRINS

1. Text Highlight

1.1. Enunț

Chiar dacă scrierea codului poate părea câteodată o acțiune ușoară, aceasta implică multe cunoștințe teoretice cât şi foarte multă practică, mai ales în scrierea de programe complexe.

Astfel, pentru a veni în ajutorul utilizatorilor, s-a creat highlighting-ul pe text, ceea ce înseamnă că vor exista cuvinte cheie precum ***"for", "while" , "int" , "float"*** care vor fi scoase în evidență pentru o înțelegere mai ușoară a codului. În această problemă ne propunem să simulăm facilitatea de TextHighlight.

Pentru a putea simula highlighting-ul pe text, vom "sublinia" keyword-urile întâlnite pe linia următoare. Sublinierea unui keyword se va face folosind caracterul '_' (underline) pentru fiecare caracter al cuvântului respectiv (incluzând spațiile în cazul keyword-urilor formate din mai multe cuvinte), iar pentru restul caracterelor care nu fac parte din cuvintele cheie se va folosi ' ' (spațiu).

Atenţie! Deoarece multe programe permit scrierea de secvențe cheie formate din mai multe cuvinte cu un număr variabil de spații între ele, pentru problema noastră va fi luat în calcul şi acest caz!

Evitați, totuşi, să scrieți cod în acest fel, deoarece devine greu de înțeles si trebuie respectat un coding style.

Pentru această temă vom considera drept cuvinte cheie (keywords) doar următoarele cuvinte şi secvenţe de două cuvinte: ***first of, for, for each, from, in, is, is a, list of, unit, or, while, int, float, double, string.***

1.2. Date de intrare

- N - numărul de linii introduse
- N linii de câte 100 de caractere maxim, reprezentând textul pe care va fi aplicat highlighting-ul

1.3. Date de ieșire

- Textul cu highlighting-ul corespunzător

1.4. Restricții și precizări

- N va fi un număr natural valid, între 0 şi 100
- Textul va fi citit linie cu linie
- Vectorul cu cele N linii va fi alocat dinamic
- Pentru fiecare linie se vor aloca exact atâtea caractere câte sunt necesare pentru stocarea acelei linii
- Soluţiile fără alocare dinamică vor fi depunctate corespunzător
- Cuvintele cheie date vor fi stocate în unul sau mai mulţi vectori, nu neapărat alocaţi dinamic.
- **Important**! Definiţi şi implementaţi **funcţii** auxiliare. Soluţiile scrise complet sau aproape complet în **main** nu vor fi luate în considerare! Nu se folosesc varaibile globale!

1.5. Testare și punctare

- Punctajul maxim este de **30** de puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **problema1.c**

1.6. Exemple

<table>
<tr><th>Input</th><th>Explicații</th></tr>
<tr><td>4
for each number from 3 to 100
divide to first
of list of divisors of number that is a number
different than 1 or 2</td><td rowspan="3">Keyword-urile care vor fi subliniate sunt:
• ‘for each’ de pe prima linie (se poate observa că a fost subliniat, deși sunt mai multe spații între cele două cuvinte) și ‘from’
• ‘list of’ de pe linia trei
• ‘is a’ de pe linia trei
• ‘or’ de pe linia patru

Observație! – Deși “first of” este un keyword, acesta NU a fost subliniat, deoarece “first” și “of” se află pe linii diferite.</td></tr>
<tr><th>Output</th></tr>
<tr><td>for each number from 3 to 100
___________ ____
divide to first

of list of divisors of number that is a number
  _______ ____
different than 1 or 2

  __</td></tr>
</table>

2. Autocomplete

2.1. Enunț

Autocomplete-ul este o funcționalitate des folosită. Deși aceasta ajută în teorie la economisirea timpului, poate genera și greșeli, depinzând de modul în care este folosită.

Cu timpul, mare parte dintre probleme au fost rezolvate și autocomplete-ul ajută aproape în fiecare caz, printre cele mai utile și folosite fiind autocomplete-ul căutării pe Google. Acest autocomplete este totuși destul de complex, folosind atât căutările precedente ale utilizatorului curent, a celorlalți utilizatori, keyword-urile paginilor existente pe Google, cât și cele mai populare căutări.

Astfel, în această problemă ne propunem să implementăm un autocomplete care pornește de la un anumit număr de cuvinte (un mini-dicționar) și care se actualizează odată cu introducerea unor cuvinte noi sau folosirea celor recente.

Fiecare cuvânt din dicționar va avea la început prioritatea 0 și va fi reprezentat sub forma aceasta:

```
struct dictionary_entry{
    char* word;
    int priority;
};
```

Fiecare cuvânt scris de utilizator face parte din una dintre următoarele 3 categorii:

- **cuvânt** -- care nu face match cu niciun cuvânt existent în dicționar, așa că va fi scris în output la fel ca în input și va fi adăugat în dicționar cu prioritatea 1, fiind prima sa apariție în text

- **cuvânt*** -- care se potrivește cu un cuvânt de prioritate mai mare, dar utilizatorul vrea cuvântul scris de el, deși are prioritate mai mică (acest cuvânt poate exista sau nu în dicționar, caz în care ar trebui adăugat) -> se va afișa **cuvânt** şi ii va creşte prioritatea

- **cuv** -- care se potrivește cu unul sau mai multe cuvinte din dicţionar (care încep cu **cuv)** şi se alege cel care are prioritatea cea mai mare. De asemenea, prioritatea cuvântului ales din dicționar va crește pentru cuvântul selectat.

  Observaţie: Dacă avem cuvinte în dicţionar cu aceeaşi prioritate şi căutăm o potrivire pentru un cuvânt dat, se va alege cuvântul considerat ca fiind cel mai aproape in ordine lexicografică.

  Spre exemplu, căutăm potrivire pentru „abc” şi avem două posibilităţi în dicţionar: ”abcde” cu prioritatea 2 si „abce” tot cu prioritatea 2. Vom alege cuvântul „abcde”, deoarece este mai mic decât „abce”.

2.2. Date de intrare

- N – numărul inițial de cuvinte din dicționar, care va fi folosit la alocarea dinamică inițială a dicţionarului
- N cuvinte – cuvintele din dicționarul inițial
- M – numărul de cuvinte introduse de utilizator pentru autocomplete
- M cuvinte – cuvintele introduse de utilizator pentru autocomplete

2.3. Date de ieșire

- Textul rezultat in urma autocomplete-ului (cuvintele rezultate, cu câte un spațiu între ele)

2.4. Restricții și precizări

- $0 < N \leq 5000$
- $0 < M \leq 5000$
- **Folosiți alocarea şi realocarea dinamică pentru dicționarul de cuvinte, astfel încât să nu existe memorie nefolosită!**
- **Fiecare cuvânt din dicţionar va avea alocată exact atâta memorie cât este necesară pentru el.**
- Soluțiile care nu folosesc alocarea dinamică vor fi depunctate.
- Pentru cuvintele introduse pentru autocomplete **NU** este necesar să folosiţi alocare dinamică!
- Caracterele (, . : ! ?) vor fi considerate cuvinte de sine stătătoare și vor fi afișate așa cum se citesc
- Toate cuvintele conțin litere mici ale alfabetului englez.
- Fiecare cuvânt introdus are maxim 20 de caractere
- **Important**! Definiţi şi implementaţi **funcţii** cel puţin pentru **căutări** în dicţionar, **adăugare** cuvânt nou în dicţionar, etc. Soluţiile scrise complet sau aproape complet în **main** nu vor fi luate în considerare!
  - Nu se folosesc variabile globale!

2.5. Testare și punctare

- Punctajul maxim este de **30** de puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **problema2.c**

2.6. Exemple

<table>
<tr><th>Input</th><th>Explicație</th></tr>
<tr><td>8

arrow discussed hi interesting was divorced interview investment

13

his inv w di in* an interv , it was rather interes .</td><td rowspan="3">În dicționar se află cuvintele arrow discussed hi interesting was divorced inverview investment. Toate au prioritate 0 în acest moment.

his – cuvânt nou ce nu corespunde niciunui cuvânt din dicționar, este afișat ca în input și adăugat în dicționar cu prioritatea 1

inv – singura potrivire este investment. Se afișează investment, dicționarul nu se modifică, iar prioritatea cuvântului „investment” crește la 1.

w – singura potrivire este was. Se afișează was, dicționarul nu se modifică, iar prioritatea cuvântului “was” crește la 1.

di – are două potriviri “discussed”, respectiv “divorced”. Ambele au aceeași prioritate, deci va fi ales cuvântul cel mai mic lexicografic. Se va afișa discussed și i se va incrementa prioritatea la 1.

in* - cuvântul cu prioritatea mai mare care se potrivește este “investment”. Totuși, utilizatorul nu vrea cuvântul recomandat, ci pe cel introdus de el. Se va afișa in (se va elimina * din output)

an – cuvânt nou, se adaugă în dicționar cu prioritatea 1

interv – singura potrivire este interview, care va avea de acum prioritatea 1

it – cuvânt nou, se adaugă în dicționar cu prioritatea 1

wa – singura potrivire este was, căruia îi va crește prioritatea la 2

rather – cuvânt nou, se adaugă în dicționar cu prioritatea 1

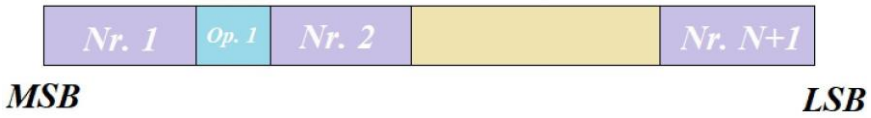
interes – singura potrivire este interesting, căruia îi va crește prioritatea la 1</td></tr>
<tr><th>Output</th></tr>
<tr><td>his investment was discussed in an interview , it was rather interesting .</td></tr>
</table>

3. Calculator pe biţi

3.1. Enunț

A treia funționalitate implementată este un calculator de operații pe biți. Acesta va fi foarte simplist, suportând până la 4 operații consecutive, pe numere de 4 biți.

Numerele și operațiile vor fi extrase dintr-un unsigned int, dat de la tastatura, număr care va fi dat sub forma:



Numerele sunt reprezentate pe câte 4 biți, iar operațiile pe câte 2 biți, operațiile fiind codificate astfel:

Operatie	Codificare
Adunare	00
Interschimbare	01
Rotație la stânga	10
Xor	11

Adunare

Rezultatul acestei operații va fi suma celor două numere pe 4 biți. Spre exemplu, după aplicarea operației de adunare pentru 0001 și 0101, va rezulta 0110.

Interschimbare

La aplicarea acestei operații se va interschimba bitul de pe pozița p1 cu bitul de pe poziția p2 pentru primul număr, unde p1 reprezintă primii 2 biți din al doilea număr, iar p2 reprezintă ultimii doi biți din al doilea număr. De exemplu, dacă se aplică operația de interschimbare pentru 0001 și 1100, rezultatul va fi 1000. Bitul de pe poziția 11 = 3 (ultimul bit din primul număr), va fi interschimbat cu bitul de pe poziția 00 = 0 (primul bit din primul număr).

Rotație la stânga

Dacă se aplică această operație pe două numere, nr1, respectiv nr2, rezultatul va fi nr1 rotit la stânga de nr2 ori. De exemplu, dacă se aplică operația de rotație pe 1011 și 0101 = 5, rezultatul va fi 0111.

XOR

Rezultatul acestei operații va fi aplicarea operatorului XOR celor două numere pe 4 biți. Spre exemplu, după aplicarea operației XOR pentru 1011 și 0101, va rezulta 1110.

3.2. Date de intrare

- N – numărul de operații ($0 \leq N \leq 4$)
- M – numărul din care sunt extrași N*6 + 4 biți de la LSB spre MSB, care reprezintă N+1 numere pe 4 biți și cele N operații pe 2 biți care se vor executa consecutiv

3.3. Date de ieșire

Rezultatul aplicării operațiilor, așa cum sunt specificate în enunț.

3.4. Restricții și precizări

- MSB = most significant bit, LSB = least significant bit
- Rezultatele operaţiilor pot avea doar 4 biți. Astfel, dacă avem de calculat 1000 +1001, rezultatul va fi 0001
- Ținând cont de restricțiile date pentru valoarea lui N, cele N+1 numere si N operații vor putea fi scrise întotdeauna într-un unsigned int, acesta având 32 de biți, iar operațiile și numerele nu pot depăși în total 28 de biți
- Dacă, spre exemplu, am avea un număr care se reprezintă pe mai mult de N*6 + 4 biți (dar în limita celor 32 de biți), vor fi extrași doar cei N*6 + 4 biți, restul neavând importanță. Astfel, pentru N = 1 și M = 3035218, se vor extrage doar ultimii 10 biți: 0001 01 0010
- **Important!** Definiţi şi implementaţi **funcţii** auxiliare. Soluţiile scrise complet sau aproape complet în **main** nu vor fi luate în considerare!
- Important! În implementarea acestui calculator pe biți, veți folosi un vector de pointeri la funcțiile care implementează cele 4 operaţii posibile ! O implementare care nu folosește pointeri la funcții va fi depunctată!
- Nu se folosesc varaibile globale!
- Pentru operațiile de adunare și xor sunt premise doar operații pe biți, astfel dacă doar se va folosi operatorul xor în calculul operației cu codificarea 11, aceasta nu va fi punctată.

3.5. Testare și notare

- Punctajul maxim este de 30 de puncte.

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **problema3.c**

3.6. Exemplu

Input	Output	Explicație
3 3035218	2	$3035218_{(10)}$ = ... 1011 10 0101 00 0001 01 0010 $_{(2)}$ **1.** 1011 rotit la stânga cu 0101 biți, adică 5 biți este 0111 **2.** Rezultatul primei operatii, 0111, este adunat cu 0001, iar rezultatul este 1000 **3.** Rezultatul celei de-a doua operații este 1000, aplicându-se operația de interschimbare. Se vor interschimba biții de pe poziția 0 și 2. Astfel rezulttul va fi 0010, adică 2.

4. WORDLE

4.1. Enunț

Vă propunem să realizați o implementare simplă, cu interfață grafică, a jocului Wordle folosing biblioteca ***ncurses***.

Wordle este un joc online ce s-a bucurat de o popularitate sporită în ultimul an. Puteți găsi o implementare a jocului pentru a vă familiariza cu regulile aici.

Regulile acestuia sunt simple. Pentru fiecare rundă, jucatorul are la dispoziție 6 incercări de a ghici un cuvânt de 5 litere. Dupa fiecare încercare, literele cuvântului scris de jucător se vor colora într-una dintre cele 3 culori (verde, galben, negru), având următoarea semnificație:

- Verde: litera se află în cuvântul ce trebuie ghicit exact pe acea poziție
- Galben: litera se află în cuvântul ce trebuie ghicit, dar pe altă poziție
- Negru: litera nu se află în cuvânt

Dacă reușește să ghicească cuvântul în cele 6 incercări, atunci jucătorul câștigă, ân caz contrar, pierde.

Cerința 1

Pentru început va trebui să construiți **grafic** ecranul de joc, în care să apară numele jocului și tabla de joc, unde utilizatorul poate introduce un cuvânt. Interfața grafică puteți să o proiectați cum doriți astfel incât să corespundă cerințelor problemei.

Cerința 2

Mai departe, urmează să implementați mecanica jocului:

- La fiecare joc nou, programul va alege cuvântul câștigător aleatoriu, din lista de cuvinte date (vezi precizări). Apoi, jucătorul trebuie să introducă câte un cuvânt, pe litere. După apăsarea tastei “Enter” se vor colora literele cuvântului conform regulilor de mai sus și se va trece la următoarea incercare.
- Dacă tasta “Enter” este apasată după un cuvânt mai scurt de 5 litere, acesta nu va fi acceptat și se va afisa un mesaj care informează jucătorul să aleagă un cuvânt din 5 litere.
- Tasta “Backspace” poate fi folosită pentru a șterge caractere din încercarea curentă.
- Dacă jucătorul ghicește întreg cuvântul, se afișează un mesaj de câștig și i se oferă posibilitatea de a începe un nou joc.
- Dacă au fost folosite toate cele 6 incercări și cuvântul nu a fost ghicit, se afișează un mesaj de joc pierdut și cuvântul care trebuia ghicit.

Cerința 3

In final urmează să implementați un meniu de control la apăsarea tastei “:”. Acesta va avea 2 opțiuni: ieșirea din aplicație și renunțarea la jocul current și începerea unuia nou.

4.2. Punctaj

Punctajul pentru acest task va fi distribuit după cum urmează:

10% - Ecranul principal cu numele jocului (Cerinta 1)
20% - Introducerea cuvintelor, funcționalitate “Backspace”
40% - Implementarea mecanicilor de joc, colorarea literelor dupa “Enter”
20% - Incheierea jocului (câștig, pierdere, reîncepere joc)
10% - Meniu de control (cerința 3)

4.3. Restricții și precizări

- Pentru implementare, veți avea nevoie de o listă de cuvinte în limba română. Trebuie să folosiți minim următoarea listă, având libertatea de a adăuga și alte cuvinte: “ arici, atent, baiat, ceata, debut, peste, fixat, hamac, harta, jalon, jucam, lacat, magie, nufar, oaste, perus, rigle, roman, sanie, scris, sonda, texte, tipar, titan, zebra, vapor, vatra”
- Aveți libertate în alegerea aspectului jocului, adică a modului în care va arăta interfața grafică a acestuia, atâta timp cât regulile de bază, descrise mai sus, sunt respectate.
- **Atenție!** Rezolvările în mod **text** (modul obișnuit de execuție a unei aplicații C de până acum) **NU** vor fi punctate!
- Cuvintele introduse de utilizator pot fi orice combinație de 5 litere, nu neaparat un cuvânt valid din limba română (de exemplu, cuvântul “abcde” se va considera valid)

4.4. Precizări *ncurses*

- Pentru obținerea bibliotecii ncurses pe o distribuție de Linux bazată pe Debian, instalați cu: apt-get install libncurses5-dev.
- **Observație**: biblioteca ncurses funcționează pe Unix!
- Detalii despre biblioteca ncurses se găsesc în următoarele materiale:
  - Documentatia oficiala
  - Scurta introducere in utilizarea bibliotecii
  - Exemple

4.5. Testare și notare

- Punctajul maxim este de 80 puncte.
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **wordle.c**
- **Nu veți avea checker pentru această problemă, ea se va testa manual si va fi prezentată și la laborator!**
- **IMPORTANT!** Această problema se va încărca în arhiva temei la fel ca și celelallte pentru a fi luată in calcul.

5. PRECIZĂRI

- Separați logica programelor în mai multe funcții, așa cum vi s-a cerut în enunțuri!
- Atentie! Sa puneti linia noua de la final de output („\n”)!
- **Nu** se vor puncta sursele în care tot programul sau apoape tot programul este scris în main!
- Toate citirile se fac de la tastatură in codul final al temei! Folosiți fișiere sau/şi redirectarea intrării/ieşirii doar pentru testele voastre intermediare!
- Datele de intrare sunt descrise pentru fiecare cerință individual și vor fi citite de la tastatură.
- Este **interzisă** folosirea variabilelor globale!
- Soluția temei va fi scrisă în ANSII C! Nu folosiți sintaxă sau instrucțiuni specifice limbajului C++.
- Fiecare problemă va fi scrisă într-un fișier sursă separat, numit așa cum s-a precizat!
- În README precizați cât timp v-a luat implementarea cerințelor și explicați, pe scurt, implementarea temei (comentariile din cod vor documenta mai amănunțit rezolvarea).
- Este recomandat ca liniile de cod și cele din fișierul README să nu depășească 80 de caractere.
- **TEMELE CARE NU AU README NU SE VOR LUA ÎN CONSIDERARE!**
- Temele sunt strict individuale. Copierea temelor va fi sancționată cu punctaj 0 pentru toţi cei care au porţiuni de cod identice!!
- Persoanele cu porțiuni de cod identice **NU** vor primi niciun punctaj pe temă.
- **NU** copiați cod de pe Internet! Se poate ajunge la situația în care doi studenți să aibă același cod, preluat de pe Internet, caz în care ambele teme vor fi punctate in TOTALITATE cu 0, deși studenții nu au colaborat direct între ei.
- Temele trimise după deadline **NU** vor fi luate în considerare.

6. CODING STYLE

Folosiți un coding style astfel încât codul să fie ușor de citit și înțeles. De exemplu:

- ✓ Dați nume corespunzătoare variabilelor și funcțiilor
- ✓ Nu adăugați prea multe linii libere sau alte spații goale unde nu este necesar:
  - nu terminați liniile în spații libere, trailing whitespaces
  - nu adăugați prea multe linii libere între instrucțiuni sau la sfârșitul fișierului
- ✓ Principalul scop al spațiilor este identarea
- ✓ Fiți consecvenți în coding style-ul ales

- ✓ Vă recomandăm să parcurgeți această resursă: https://ocw.cs.pub.ro/courses/programare-cc/coding_style
- ✓ Există programe sau extensii pentru editoare text care vă pot formata codul
- ✓ Deși vă pot ajuta destul de mult, ar fi ideal să încercați să respectați coding style-ul pe măsură ce scrieți codul

7. TRIMITEREA TEMEI

Veţi trimite o arhivă **ZIP** cu numele **EXACT** acesta: **GRUPA_Nume_Prenume_Tema2.zip**.
De exemplu: 311CC_Popescu_Maria_Tema2.zip.
Arhiva va conține următoarele fișiere:

1. README
2. Makefile
3. problema1.c
4. problema2.c
5. problema3.c
6. wordle.c

Atenţie! Veţi fi depunctaţi complet pentru formatarea incorectă a arhivei (alt nume, alt tip, alte nume pentru fişiere, alte fişiere în plus sau în minus etc.) - 0 puncte pe temă

- Pentru întrebări legate de temă se va folosi în mod exclusiv **FORUM-ul** temei, pe care vă recomandăm să îl vizitați chiar și dacă nu aveți întrebări, întrucât este posibil să aflați informații noi din întrebările puse de colegii voștri, respectiv din raspunsurile date de noi.
- Compilarea nu ar trebui să producă warning-uri (verificați prin adăugarea flagului -Wall la gcc).
- Temele trebuie să fie încărcate pe **vmchecker**. **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.
  Link vmchecker: https://vmchecker.cs.pub.ro/ui/

8. PUNCTARE

Tema 2 va avea un punctaj de maxim **1.7p** din nota finală din care **0.5p** vor fi **BONUS**, bonus care sa va lua însă in calcul pentru nota finală!

EXERCIȚIU	PUNCTAJ
Text Highlight	**30 puncte**
Autocomplete	**30 puncte**
Calculator pe biți	**30 puncte**
WORDLE	**80 puncte**
TOTAL	**170 puncte (din care 50p bonus)**

Temele vor fi prezentate asistenților în cadrul primelor două laboratoare de după deadline!

Depunctările care se pot aplica:

- - 10 puncte pentru explicațiile din README
- - 10 puncte pentru lipsă comentarii din cod (atentie! nu trebuie comentată fiecare linie, doar ceea ce este esenţial pentru a se înţelege uşor rezolvarea – detalii la curs!)
- - 20 puncte pentru Coding Style
- Depunctare totală pentru lipsă fisier README (0 puncte pe tema)
- **Depunctare totală pentru formatarea incorectă a arhivei (alt nume, alt tip, alte nume pentru fisiere, alte fisiere in plus sau in minus)** - 0 puncte pe tema

O temă care NU compilează va fi punctată cu 0.

O temă care NU trece niciun test pe vmchecker va fi punctată cu 0.

Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.

Pentru **problema 4 – WORDLE nu veţi avea scenarii şi checker!**

În fișierul README va trebui să descrieți pe scurt soluția pe care ați ales-o pentru fiecare problemă și alte lucruri pe care le considerați utile de menționat.

9. CHECKER

- Arhiva se va trimite pe vmchecker, unde tema se va testa automat (primele 3 probleme).
- Pentru testarea locală, aveți disponibil un set de teste și un checker local (primele 3 probleme).
- Punctajul acordat pe rularea testelor este cel de pe vmchecker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă gsăsită în implementare, dacă vor considera acest lucru necesar.
- Problema 4 se va testa manual și se va prezenta la laborator!