Tema 1 - Calculator

Responsabili:

- Răzvan Nițu

Termen de predare:

- Deadline soft: **16.11.2022 23:55**
- Deadline hard: **22.11.2022 23:55**

Cerință

Pentru aceasta tema va trebui sa implementati un calculator care sa faca operatii aritmetice. Programul citeste de la tastatura o expresie aritmentica si afiseaza rezultatul acestei expresii. Formatul de citire a expresiei de la tastatura este urmatorul: se citeste un numar N, apoi N numere reale, iar in final se citesc N-1 operatii. O operatie este reprezentata de un caracter si poate avea valorile: '+', '-', '*', '/'. Astfel, urmatorul input:

```
4    // N = 4

3    // operand 1

3    // operand 2

4    // operand 3

5    // operand 4

+*/  // N-1 = 3 operatii
```

Va fi interpretat ca: “3+3*4/5”.

Task 1

Pentru prima cerinta va trebui sa implementati citirea unei expresii si evaluarea acesteia, neglijand precedenta operatorilor. Astfel, pentru expresia “3+3*4/5”, veti evalua operatiile in ordinea in care apar: 3+3*4/5 6*4/5 = 24/5 = 4.8 . Afisarea rezultatului va avea si o linie noua (adica “printf(”%f\n”, result)”, *NU* “printf(”%f”, result)”).

Task 2

Pentru a doua cerinta va trebui sa implementati citirea unei expresii si evaluarea acesteia tinand cont si de precendenta operatorilor. Astfel, pentru expresia “3+3*4/5”, veti evalua operatiile tinand cont de faptul ca operatiile '*' si '/' au precedenta mai mare fata de '+' si '-'. In consecinta, pentru acesta sarcina, rezultatul expresiei anterioare va fi: 3+3*4/5 = 3+12/5 = 3+2.4 = 5.4 .

Task 3

Pentru a 3-a cerinta va trebui sa implementeti un nou operator, operatorul `#`. Operatorul `#` este definit astfel: a#b = (a+b)*(a+b). Operatorul `#` are precedenta mai mare ca `+`/`-`, dar mai mica ca `*`/`/`. Astfel, pentru aceasta cerinta, rezultatul expresiei "1+2#3*4" va fi: 1+2#3*4 = 1+2#12 = 1 + (2 + 12)*(2 + 12) = 1 + 196 = 197.

Precizari

Checker-ul se asteapta ca fiecare task sa fie separat intr-un fisier separat. Astfel, este necesar ca Makefilul pe care il puneti in arhiva alaturi de fisierele sursa sa creeze binarele cu numele **task1**, **task2**, **task3**. Makefile-ul oferit alaturi de checker respecta aceasta cerinta, astfel, in cazul in care modificati Makefile-ul, asigurati-va ca binarele finale vor avea numele corespunzatoare.
Dacă aveți nelămuriri, puteți să ne contactați pe forumul dedicat temei de casă nr. 1 sau pe canalul Temei 1.
La orice întrebare vom răspunde în maxim 24 de ore.
Nu se acceptă întrebări în ultimele 24 de ore înainte de deadline.

Trimitere temă

Tema va fi trimisă folosind vmchecker, cursul **Programarea Calculatoarelor (CB & CD)**.
Toate temele sunt testate în mod automat pe VMChecker. Autentificarea se face folosind numele de utilizator și parola de pe moodle / Teams.
Din meniul *drop-down* selectați cursul corespunzător; în cazul de față: **Programarea Calculatoarelor (CB & CD)**. În meniul *didebar*, din partea stângă a paginii, selectați tema pentru care veți face submisia.
Arhiva temei se va încărca pe checker folosind formularul de submisie din tabul **Trimitere solutii**.
Rezultatele vor fi disponibile în tabul **Rezultate**.
Citiți cu atenție informațiile afișate în **Rezultate** pentru a vă asigura că tema a fost rulată cu succes; o eroare comună este dată de faptul că conținutul arhivei nu respectă structura dorită (ex. fișierele sunt într-un alt director).
Punctajul final al temei este afișat la finalul informațiilor afișate în **Rezultate**.
Conținutul arhivei va fi următorul:

1. Fișierele **.c**, **.h** (dacă este cazul) care conține implementarea temei.
2. Fișierul **Makefile**.
3. Un fișier README în care descrieți rezolvarea temei.

1. Arhiva trebuie să fie de tipul zip.
2. Makefile-ul și testele vor fi cele din aceasta arhiva: calculator.zip
3. Puteți utiliza regula archive din fișerul Makefile pentru a vă genera arhiva zip. Aceasta va adăuga fișierele Makefile, README și orice fișier .c și .h din directorul curent.

```
make archive
```

Nu includeti fisierele checkerului in arhiva voastra. **Nu folositi Makefile.checker** pe post de Makefile in arhiva voastra: asta va duce la recursivitate infinita pe vmchecker. Puteti sa folositi direct makefile-ul prezent in arhiva (**Makefile, nu Makefile.checker**).

In cazul in care testele va trec local, insa pica pe vmchecker cel mai probabil aveti o sursa de "undefined behavior in cod". Pentru a va asigura ca scapati de aceste probleme, compilati cu flagul de compilare `-Wall` si rezolvati toate warning-urile.

Listă depunctări

Lista nu este exhaustivă.

- O temă care nu compilează și nu a rulat pe **vmchecker** nu va fi luată în considerare
- O temă care nu rezolvă cerința și trece testele prin alte mijloace nu va fi luată în considerare
- **NU acceptăm teme copiate**. În cazul unei teme copiate se scade punctajul aferent temei din punctajul total.
- [-20.0]: Nerezolvarea tuturor erorilor și warningurilor de coding style