Paradigme de Programare
Tema 3 - Prolog

Termen de predare: **29.05.2016** (soft)
05.06.2016 (hard)

Ultima actualizare: 20.05.2016 (v.2)

v.2 : *a fost adaugata erata; deocamdata doar despre unul dintre testele publice*
v.1 : *versiunea initiala*

1 Inca un limbaj IMP

Se da un limbaj de programare similar cu cel din Tema 2 (numit tot "IMP"), definit de urmatoarea gramatica:

```
<expr>   ::= <expr> <op> <expr> | <symbol> | <value>
<symbol> ::= [a-zA-Z]+
<value>  ::= [1-9][0-9]* | 0
<op>     ::= + | - | * | == | <
<prog>   ::= <symbol> = expr ;
           | <prog> <prog>
           | if(<expr>) then {<prog>} else {<prog>}
           | while(<expr>) {<prog>}
           | return <expr> ;
           | ;
```

Nu este necesar ca ultima instructiune din codul sursa sa fie de tip *return* pentru a avea un program valid, atat timp cat ultima instructiune *executata* este de acest tip. De exemplu, programul urmator este valid si are ca rezultat valoarea 3.

```
a = 1;
if(a < 5) then {
    return 3;
}
else {
;
}
b = a + 2;
c = 5 * b;
```

Rezultatul este invalid doar atunci cand nu mai este executat blocul asociat instructiunii *if*, deoarece in acest caz se ajunge la sfarsitul programului fara a fi executat vreun *return*.

Toate variabilele sunt implicit globale si nu pot fi folosite fara a le initializa in prealabil. Orice modificare adusa unei variabile intr-un bloc al programului este ulterior vizibila oriunde, si nu se limiteaza doar la regiunea respectiva (de exemplu interiorul unui bloc *else*). Prin urmare, programul urmator are 11 ca rezultat.

```
if(1) then {
    a = 10;
}
else {
;
}
b = a + 1;
return b;
```

Pe de alta parte, daca instructiunea *if* de mai sus primeste o conditie ce nu este adevarata, programul devine invalid pentru ca variabila a ajunge sa fie folosita fara a fi fost initializata anterior.

De asemenea, pentru ca rezultatul unui program sa fie valid, trebuie ca intregul program sa fie corect din punct de vedere *sintactic*. De exemplu, programul urmator va produce o eroare, chiar daca instructiunea *return* apare inainte de linia problematica.

```
return 1;
b = b / 2;
```

Erorile *semantice* nu conteaza pana la evaluarea efectiva a declaratiei in care apar. Singura eroare semantica posibila este folosirea unei variabile neinitializate. Programul urmator este valid si are care rezultat valoarea 1.

```
return 1;
b = X + 2;
```

Se considera ca rezultatul celor doua operatii logice ('<' si '==') este numeric, si poate avea doua valori posibile: 1 (adevarat) sau 0 (fals). Operatiile au prioritati diferite. In ordine descrescatoare, acestea sunt: *, + si -, <, ==. In aceste conditii, rezultatul programului urmator este 0.

```
return 1 + 2 * 3 < 5 == 1;
```

Prima data se evalueaza inmultirea, apoi adunarea, si se obtine $7 < 5 == 1$. In continuare se evalueaza operatia <, ce are rezultatul 0, si in final expresia $0 == 1$, care duce tot la valoarea 0.

Conditia asociata instructiunilor *if* sau *while* este falsa atunci cand expresia asociata se evalueaza la 0, si adevarata in celelalte cazuri.

Cuvintele cheie (*if*, *then*, *else*, *while* si *return*) nu au voie sa apara pe post de <*symbol*>. In caz contrar, trebuie semnalata o eroare.

2 Schelet de cod

In cadrul fisierului "tema3.pl" veti gasi un parser ajutator si definitia predicatului *parseInput*, care este folosit pentru a evalua rezultatul prelucrarii unui fisier sursa.

Parserul ajutator *lparse* primeste un string, pe care il transforma intr-o lista de tokens. Acestea pot fi de forma [a-zA-Z0-9]+, ';', '<', '+', '-', '*', '(', ')', '{', '}', '=' sau '=='. Orice altceva (inclusiv spatiile libere) este ignorat.

Practic, acest parser transforma string-ul respectiv intr-o lista de elemente constituente ale limbajului IMP, cu mentiunea importanta ca nu distinge intre valori si simboluri; ambele pot fi cuprinse de un token de tip [a-zA-Z0-9]+. Ramane in sarcina voastra sa va asigurati ca simbolurile si valorile sunt conforme cu definitia limbajului.

Pentru a vedea mai bine cum arata rezultatul parserului ajutator, puteti evalua expresia:

read_file_to_string('C:\\tema3pp\\tests\\public\\t001.in', S, []),
lparse(S,L), *write*(L), *fail* .

Calea de mai sus depinde, bineinteles, de locul unde se afla directorul ce contine testele publice (sau orice alte fisiere pe care doriti sa le parsati), si numele fisierului.

Dupa cum se poate observa din scheletul de cod, predicatul *parseInput(F,R)* incepe prin a citi intr-un string continutul fisierului F, pe care il transforma intr-o lista de tokens cu ajutorul lui *lparse*, si in cele din urma face apel la *parseInputAux* pentru a obtine rezultatul.

3 Cerinte

Se cere implementarea predicatului *parseInputAux*, astfel incat *parseInput* sa obtina valoarea corecta asociata evaluarii fisierului sursa primit ca parametru. Un program IMP poate duce la urmatoarele valori:

- 'e', daca s-a intalnit o eroare sintactica sau semantica, avand in vedere cele mentionate in sectiunea introductiva. Nu trebuie aduse precizari suplimentare cu privire la tipul erorii.
- 'x', daca programul a fost executat fara erori, insa am ajuns la sfarsitul lui fara a intalni vreo instructiune *return*.
- O valoare numerica, obtinuta in urma executiei unei instructiuni *return*.

In primele doua cazuri de mai sus, trebuie folosite doar valori lower-case. Nu modificati definitia lui *parseInput*.

4 Testare

Puteti gasi testele publice in directorul "tests\public". Fisierele ".out" contin rezultatul asteptat de la fiecare test. Fisierul "errors.txt" contine motivul pentru care testele trebuie sa raporteze prezenta unor erori.

Pentru executia fiecarui program se vor aloca maxim trei secunde. Acest interval este mult mai mare decat durata preconizata de executie a testelor.

Atat testele publice cat si cele private contin un subset de "teste simple". Acestea constau intr-o singura instructine de tip *return* aplicata unei expresii ce poate contine doar valori si operatii de adunare, scadere si/sau inmultire. De exemplu, primele trei teste publice intra in aceasta categorie.

5 Trimitere

Tema trebuie trimisa intr-o arhiva ".zip" care sa contina doar doua fisiere (fara vreun director intermediar): scheletul de cod completat ("tema3.pl") si un fisier README. Numele arhivei trebuie sa fie de forma "grupa_nume_prenume.zip".

6 Punctaj

Tema valoreaza 1 punct din nota finala, impartit in felul urmator:

- 0.3 puncte pentru trecerea testelor simple.
- 0.7 puncte pentru trecerea celorlalte teste.

Temele trimise dupa termenul soft sunt penalizate cu 0.05 puncte/zi. Cele trimise dupa termenul hard nu mai sunt luate in considerare.

Erata

- In prima varianta in care au fost incarcate pe site testele publice, fisierul **'t008.in'** contine cateva erori ce n-ar fi trebuit sa fie prezente; am adaugat versiunea corecta si am actualizat arhiva respectiva.