ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ
ΕΡΓΑΣΙΑ Β1

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:ΠΙΚΡΙΔΑΣ ΜΕΝΕΛΑΟΣ 141291
ΟΝΟΜΑΤΕΠΩΝΥΜΟ:ΣΤΑΘΑΚΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 161041
ΤΜΗΜΑ: ΠΕΜΠΤΗ 15:00-17:00
ΑΡΙΘΜΟΣ ΟΜΑΔΑΣ:12

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Περιεχόμενα

ΕΞΑΝΤΗΤΙΚΟΙ ΕΛΕΓΧΟΙ ΣΥΝΤΑΚΤΙΚΟΥ ΑΝΑΛΥΤΗ

Έλεγχος εντολών υπολογισμού πράξεων (δύο αριθμούς-μεταβλητές)

Εντολή: `(+ 5 3)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 22
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 41
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 77
Reducing stack by rule 5 (line 45):
   $1 = token PARENTHESI1 ()
   $2 = token PLUS ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
8
```

```
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της πρόσθεσης των δύο ακεραίων).

Εντολή: `(- 7 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 11
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 28
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 53
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 113
Reducing stack by rule 17 (line 58):
   $1 = token PARENTHESI1 ()
   $2 = token MINUS ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της αφαίρεσης των δύο ακεραίων).

Εντολή: `(* 4 3)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MUL ()
Shifting token MUL ()
Entering state 9
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 24
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 45
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 89
Reducing stack by rule 29 (line 71):
   $1 = token PARENTHESI1 ()
   $2 = token MUL ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
12
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ του πολλαπλασιασμού των δύο ακεραίων).

Εντολή: `(/ 5 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token DIV ()
Shifting token DIV ()
Entering state 10
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 26
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 49
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 101
Reducing stack by rule 41 (line 84):
   $1 = token PARENTHESI1 ()
   $2 = token DIV ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της διαίρεσης των δύο ακεραίων).

Εντολή: `(+ ?var 3)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 23
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 43
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 83
Reducing stack by rule 6 (line 46):
   $1 = token PARENTHESI1 ()
   $2 = token PLUS ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
3
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται πρόσθεση ακεραίου με την τιμή μιας μεταβλητής).

Εντολή: `(- 7 ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 11
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 28
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 54
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 116
Reducing stack by rule 19 (line 60):
   $1 = token PARENTHESI1 ()
   $2 = token MINUS ()
   $3 = token INTCONST ()
   $4 = token VARIABLE ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
7
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται αφαίρεση ακεραίου με την τιμή μιας μεταβλητής).

Εντολή: `(* ?var ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MUL ()
Shifting token MUL ()
Entering state 9
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 25
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 48
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 98
Reducing stack by rule 32 (line 74):
   $1 = token PARENTHESI1 ()
   $2 = token MUL ()
   $3 = token VARIABLE ()
   $4 = token VARIABLE ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται πολλαπλασιασμός με τις τιμές δύο μεταβλητών).

Εντολή: `(/ ?var 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token DIV ()
Shifting token DIV ()
Entering state 10
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 27
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 51
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 107
Reducing stack by rule 42 (line 85):
   $1 = token PARENTHESI1 ()
   $2 = token DIV ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται διαίρεση ακεραίου με την τιμή μιας μεταβλητής).

Λάθος εντολές

Εντολή: `(+ OIUY 9)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token:
O BUFFER TWRA EXEI MESA: OIUY
Error: invalid character
Next token is token $undefined ()
Error: syntax error
Error: popping token PLUS ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token $undefined ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής δεν είναι επιτυχής διότι αντί για μεταβλητή ή ακέραιο αριθμό δίνεται όρισμα οπότε είναι αδύνατη η εκτέλεση της πρόσθεσης.

Εντολή: `(* ORISMA 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MUL ()
Shifting token MUL ()
Entering state 9
Reading a token:
O BUFFER TWRA EXEI MESA: ORISMA
Error: invalid character
Next token is token $undefined ()
Error: syntax error
Error: popping token MUL ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
```

```
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token $undefined ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής δεν είναι επιτυχής διότι αντί για μεταβλητή ή ακέραιο αριθμό δίνεται όρισμα οπότε είναι αδύνατη η εκτέλεση του πολλαπλασιασμού.

Αντίστοιχα, η εντολή `(/ POIUY ?v)` δεν είναι συντακτικά ορθή δίοτι δεν μπορεί να εκτελεστή διαίρεση με την τιμή μιας μεταβλητής και με ένα όρισμα.

Εντολή: `(- ?JHUG ;JHG)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 11
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 29
Reading a token: This is a comment
Next token is token PARENTHESI2 ()
Error: syntax error
Error: popping token VARIABLE ()
Stack now 0 1 3 11
Error: popping token MINUS ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token PARENTHESI2 ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής δεν είναι επιτυχής διότι είναι αδύνατη η εκτέλεση της αφαίρεσης με την τιμή μιας μεταβλητής και ενός σχολίου.

Έλεγχος εντολών υπολογισμού πράξεων (τρείς αριθμοί-μεταβλητές)

Εντολή: `(+ 5 3 8)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 22
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 41
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 78
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 161
Reducing stack by rule 9 (line 49):
   $1 = token PARENTHESI1 ()
   $2 = token PLUS ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token INTCONST ()
   $6 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
16
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται πρόσθεση τριών ακεραίων).

Εντολή: `(- 7 6 1)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 11
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 28
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 53
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 114
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 185
Reducing stack by rule 21 (line 62):
   $1 = token PARENTHESI1 ()
   $2 = token MINUS ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token INTCONST ()
   $6 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται αφαίρεση τριών ακεραίων).

Εντολή: `(* 4 3 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token MUL ()
Shifting token MUL ()
Entering state 9
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 24
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 45
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 90
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 169
Reducing stack by rule 33 (line 75):
   $1 = token PARENTHESI1 ()
   $2 = token MUL ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token INTCONST ()
   $6 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
72
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται πολλαπλασιασμός τριών ακεραίων).

Εντολή: `(/ 5 6 3)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token DIV ()
Shifting token DIV ()
Entering state 10
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 26
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 49
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 102
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 177
Reducing stack by rule 45 (line 88):
   $1 = token PARENTHESI1 ()
   $2 = token DIV ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token INTCONST ()
   $6 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται διαίρεση τριών ακεραίων).

Εντολή: `(+ ?var 3 ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 23
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 43
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 85
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 166
Reducing stack by rule 14 (line 54):
   $1 = token PARENTHESI1 ()
   $2 = token PLUS ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token VARIABLE ()
   $6 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
3
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ γίνεται πρόσθεση τριών τιμών από μεταβλητές).

Οι δύο παρακάτω εντολές, όταν γίνει εισαγωγή στον συντακτικό αναλυτή ο έλεγχος και η εκτέλεση είναι επιτυχής διότι υπολογίζουν τις πράξεις αφαίρεσης και πολλαπλασιασμού για δύο μεταβλητές και έναν ακέραιο αντίστοιχα.

```
(- 7 ?var ?var)
(* ?var ?var 5)
```

Λάθος εντολές

Εντολή: `(/ ?var 6 ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token DIV ()
Shifting token DIV ()
Entering state 10
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 27
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 51
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 109
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 182
Reducing stack by rule 50 (line 93):
   $1 = token PARENTHESI1 ()
   $2 = token DIV ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token VARIABLE ()
   $6 = token PARENTHESI2 ()
```

Στην περίπτωση αυτή η εντολή είναι συντακτικά σωστή αλλά επειδή οι τιμές των μεταβλητών είναι 0 και η πράξη είναι διαίρεση είναι αδύνατο να εκτελεστεί.

Εντολή: `(+ ?var ORISMA 1)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 8
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 23
Reading a token:
O BUFFER TWRA EXEI MESA: ORISMA
Error: invalid character
Next token is token $undefined ()
Error: syntax error
Error: popping token VARIABLE ()
Stack now 0 1 3 8
Error: popping token PLUS ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token $undefined ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής είναι ανεπιτυχής διότι εκτός από μεταβλητή και αριθμό, περίεχει και όρισμα. Αντίστοιχα οι παρακάτων εντολές δεν εκτελούνται γιατί έχουν και ορίσματα ή σχόλια ή χαρακτήρες που δεν αναγνωρίζονται από την mini-CLIPS.

```
(- 5 ?var ;JHG)
(* 6 ;JHGF #$$%^)
(/ ?v %&* ORISMA)
```

Έλεγχος εντολών σύγκρισης (με τον τελεστή =)

Εντολή: `(= 5 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 31
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 62
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 137
Reducing stack by rule 53 (line 97):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token INTCONST ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακεραίων και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ είναι ίσοι οι αριθμοί άρα έχουμε 1).

Εντολή: `(= ?var 2)`
Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 32
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 65
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 147
Reducing stack by rule 54 (line 98):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση ακεραίου με μια μεταβλητή και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Εντολή: `(= 2 ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 31
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 63
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 140
Reducing stack by rule 55 (line 99):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token INTCONST ()
   $4 = token VARIABLE ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση ακεραίου με μια μεταβλητή και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Εντολή: `(= ?var ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 32
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 66
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 150
Reducing stack by rule 56 (line 100):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token VARIABLE ()
   $4 = token VARIABLE ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο μεταβλητών και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ είναι ίσες οι τιμές άρα έχουμε 1).

Εντολή: `(=(+ 3 6) 4)` Σημείωση: Αυτή και οι παρακάτω εντολές στους πρώτους 3 χαρακτήρες δεν πρέπει να έχουν κενό για να αναγνωριστούν από τον συντακτικό αναλύτη.

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 30
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 57
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 125
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 193
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 239
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 292
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 359
Reducing stack by rule 65 (line 110):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token PARENTHESI1 ()
   $4 = token PLUS ()
   $5 = token INTCONST ()
   $6 = token INTCONST ()
   $7 = token PARENTHESI2 ()
   $8 = token INTCONST ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
```

```
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακέραιων τιμών (η μια προκύπτει από την πρόσθεση δύο ακέραιων) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Εντολή: `(=( - 3 6) 4)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 30
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 60
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 131
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 205
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 251
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 316
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 383
Reducing stack by rule 66 (line 111):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token PARENTHESI1 ()
   $4 = token MINUS ()
```

```
   $5 = token INTCONST ()
   $6 = token INTCONST ()
   $7 = token PARENTHESI2 ()
   $8 = token INTCONST ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακέραιων τιμών (η μια προκύπτει από την αφαίρεση δύο ακέραιων) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Κατά αντιστοιχεία οι επόμενες δύο εντολές εκτελούνται ορθά και έχουν αναμενόμενα αποτελέσματα.

```
(=(* 3 6) 4)
(=(/ 3 6) 4)
```

Εντολή: `(=9(+ 4 5))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 31
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
```

```
Entering state 61
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 133
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 209
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 255
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 324
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 391
Reducing stack by rule 97 (line 143):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token INTCONST ()
   $4 = token PARENTHESI1 ()
   $5 = token PLUS ()
   $6 = token INTCONST ()
   $7 = token INTCONST ()
   $8 = token PARENTHESI2 ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακέραιων τιμών (η μια προκύπτει από την πρόσθεση δύο ακέραιων) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ είναι ίσες οι τιμές άρα έχουμε 1).

Εντολή: `(=(- ?m 6) 4)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 30
Reading a token: Next token is token MINUS ()
Shifting token MINUS ()
Entering state 60
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 132
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 207
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 253
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 320
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 387
Reducing stack by rule 70 (line 115):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token PARENTHESI1 ()
   $4 = token MINUS ()
   $5 = token VARIABLE ()
   $6 = token INTCONST ()
   $7 = token PARENTHESI2 ()
   $8 = token INTCONST ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
```

```
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακέραιων τιμών (η μια προκύπτει από αφαίρεση ακεραίου με μεταβλητή) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0). Αντίστοιχα το ίδιο ισχυεί και για την παρακάτω εντολή.

```
(=(* 3 ?v) 4)
```

Εντολή: `(=(* 3 8) ?d)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 30
Reading a token: Next token is token MUL ()
Shifting token MUL ()
Entering state 58
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 127
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 197
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 243
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 301
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 368
Reducing stack by rule 75 (line 120):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
```

```
   $3 = token PARENTHESI1 ()
   $4 = token MUL ()
   $5 = token INTCONST ()
   $6 = token INTCONST ()
   $7 = token PARENTHESI2 ()
   $8 = token VARIABLE ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο τιμών (η μια προκύπτει από πολλαπλασιασμό δύο ακεραίων και η επόμενη από μεταβλητή) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Εντολή: `(=(+ ?m ?s) ?c)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 30
Reading a token: Next token is token PLUS ()
Shifting token PLUS ()
Entering state 57
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
```

```
Entering state 126
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 196
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 242
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 299
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 366
Reducing stack by rule 93 (line 138):
   $1 = token PARENTHESI1 ()
   $2 = token EQUAL ()
   $3 = token PARENTHESI1 ()
   $4 = token PLUS ()
   $5 = token VARIABLE ()
   $6 = token VARIABLE ()
   $7 = token PARENTHESI2 ()
   $8 = token VARIABLE ()
   $9 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο τιμών (η μια προκύπτει από πρόσθεση δύο μεταβλητών και η επόμενη από μεταβλητή) και ως αποτέλεσμα επιστρέφει 0 ή 1 ανάλογα το αποτέλεσμα της σύγκρισης (εδώ δεν είναι ίσες οι τιμές άρα έχουμε 0).

Λάθος εντολές

Εντολή: `(= lhg ?var)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token:
O BUFFER TWRA EXEI MESA: lhg
Error: invalid character
Next token is token $undefined ()
Error: syntax error
Error: popping token EQUAL ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token $undefined ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής δεν είναι επιτυχής διότι δεν είναι δυνατή η σύγκριση μιας και τιμής και ενός ορίσματος.

Εντολή: `(= ?var ;LKJHGF)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Reading a token: (= ?var ;LKJHGF)
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 12
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 32
Reading a token: This is a comment
Next token is token PARENTHESI2 ()
```

```
Error: syntax error
Error: popping token VARIABLE ()
Stack now 0 1 3 12
Error: popping token EQUAL ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token PARENTHESI2 ()
Stack now 0
```

Η εκτέλεση της παραπάνω εντολής δεν είναι επιτυχής διότι δεν είναι δυνατή η σύγκριση μιας και τιμής και ενός σχολίου.

Έλεγχος εντολών σύγκρισης (με την εντολή test)

Εντολή: `(TEST ?var 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: TEST
Next token is token TEST ()
Shifting token TEST ()
Entering state 13
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 34
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 68
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 155
Reducing stack by rule 126 (line 173):
   $1 = token PARENTHESI1 ()
```

```
   $2 = token TEST ()
   $3 = token VARIABLE ()
   $4 = token INTCONST ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
0
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση ενός ακεραίου με την τιμή μιας μεταβλητής και επιστρέφει 0 ή 1 ανάλογα τις τιμές (εδώ επιστρέφει 0 γιατί δεν είναι ίσες οι τιμές).

Εντολή: `(TEST ?var1 ?var2)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: TEST
Next token is token TEST ()
Shifting token TEST ()
Entering state 13
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 34
Reading a token: Next token is token VARIABLE ()
Shifting token VARIABLE ()
Entering state 69
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 156
Reducing stack by rule 127 (line 174):
   $1 = token PARENTHESI1 ()
```

```
   $2 = token TEST ()
   $3 = token VARIABLE ()
   $4 = token VARIABLE ()
   $5 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο τιμών από μεταβλητές και επιστρέφει 0 ή 1 ανάλογα τις τιμές (εδώ επιστρέφει 1 γιατί είναι ίσες οι τιμές).

Εντολή: `(test(= 6 7))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: test
Next token is token TEST ()
Shifting token TEST ()
Entering state 13
Reading a token: Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 33
Reading a token: Next token is token EQUAL ()
Shifting token EQUAL ()
Entering state 67
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 153
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
```

```
Entering state 233
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 284
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 353
Reducing stack by rule 128 (line 176):
   $1 = token PARENTHESI1 ()
   $2 = token TEST ()
   $3 = token PARENTHESI1 ()
   $4 = token EQUAL ()
   $5 = token INTCONST ()
   $6 = token INTCONST ()
   $7 = token PARENTHESI2 ()
   $8 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
1
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται σύγκριση δύο ακεραίων και επιστρέφει 0 ή 1 ανάλογα τις τιμές (εδώ επιστρέφει 1 γιατί είναι ίσες οι τιμές).

Αντίστοιχα οι παρακάτω εντολές εκτελούνται με επιτυχία διότι γίνεται σύγκριση δύο τιμών (από μεταβλητή και ακέραιο) και επιστρέφουν 0 ή 1 ανάλογα τις τιμες.

```
(test(= ?r 7))
(test(= ?r ?z))
(test(= 5 ?z))
```

Λάθος εντολές

Εντολή: `(TEST ;KJHG 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Reading a token: (TEST ;KJHG 5)
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: TEST
Next token is token TEST ()
Shifting token TEST ()
Entering state 13
Reading a token: This is a comment
Next token is token PARENTHESI2 ()
Error: syntax error
Error: popping token TEST ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token PARENTHESI2 ()
Stack now 0
```

Η εκτέλεση της εντολής είναι ανεπιτυχής διότι δεν γίνεται σύγκριση μεταξύ μιας τιμής και ενός σχολίου ή ορίσματος. Το ίδιο ισχυεί και για την παρακάτω εντολή.

```
(TEST ?var kjhgf)
```

Έλεγχος εντολών για την ανάγνωση ακεραίων

Εντολή: `(READ 7)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Reading a token: (READ 7)
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: READ
Next token is token READ ()
Shifting token READ ()
Entering state 16
```

```
Reading a token: Next token is token INTCONST ()
Shifting token INTCONST ()
Entering state 37
Reading a token: Next token is token PARENTHESI2 ()
Shifting token PARENTHESI2 ()
Entering state 73
Reducing stack by rule 140 (line 215):
   $1 = token PARENTHESI1 ()
   $2 = token READ ()
   $3 = token INTCONST ()
   $4 = token PARENTHESI2 ()
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Reading a token: Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
7
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Η εκτέλεση της παραπάνω εντολής είναι επιτυχής διότι γίνεται είσοδος από το πληκτρολόγιο μιας τιμής και την εμφανίζει ο συντακτικός αναλυτής στην οθόνη.

Λανθασμένη εντολή

Εντολή: `(READ LKJH) //LATHOS`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση.

Αποτέλεσμα:

```
Reading a token: (READ LKJH)
Next token is token PARENTHESI1 ()
Shifting token PARENTHESI1 ()
Entering state 3
Reading a token:
O BUFFER TWRA EXEI MESA: READ
Next token is token READ ()
Shifting token READ ()
Entering state 16
Reading a token:
O BUFFER TWRA EXEI MESA: LKJH
```

```
Next token is token PARENTHESI2 ()
Error: syntax error
Error: popping token READ ()
Stack now 0 1 3
Error: popping token PARENTHESI1 ()
Stack now 0 1
Error: popping nterm program ()
Stack now 0
Cleanup: discarding lookahead token PARENTHESI2 ()
Stack now 0
```

Η mini-CLIPS δεν υποστιρίζει είσοδο κειμένου από το πληκτρολόγιο με αποτέλεσμα η εντολή να μην εκτελείται.

Έλεγχος σχολίων

Σχόλιο: `;COMMENTS?+-`

Αποτέλεσμα:

```
Reading a token: ;COMMENTS?+-
This is a comment
Next token is token NEWLINE ()
Reducing stack by rule 143 (line 220):
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Next token is token NEWLINE ()
Shifting token NEWLINE ()
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
-321462832
-> $$ = nterm program ()
Stack now 0
Entering state 1
Reading a token: Next token is token NEWLINE ()
Reducing stack by rule 143 (line 220):
-> $$ = nterm expr ()
Stack now 0 1
Entering state 6
Next token is token NEWLINE ()
Shifting token NEWLINE ()
```

```
Entering state 20
Reducing stack by rule 1 (line 38):
   $1 = nterm program ()
   $2 = nterm expr ()
   $3 = token NEWLINE ()
-321462832
-> $$ = nterm program ()
Stack now 0
Entering state 1
```

Το σχόλιο είναι δεκτό και ο συντακτικός αναλυτής συνεχίζει με τις επόμενες εντολές ή σχόλια.

Υποσημειώσεις:

*Στις παραπάνω εντολές, οι μεταβλητές δεν περιέχουν τιμή (εκτός από την εντολή read).

*Οι παραπάνω εντολές εκτέλεσης πράξεων εκτελούνται μέχρι και με τρείς μεταβλητές ή ακέραιους.

*Οι δεσμευμένες λέξεις είναι δεκτές με πεζούς και κεφαλαίους χαρακτήρες.

*Στην παρούσα εργασία η εισαγωγή των εντολών στον συντακτικό αναλυτή γίνεται με επιτυχία από το αρχείο input.txt αλλά δεν αποθηκέυονται οι ορθές εντολές στο αρχείο output.txt. Θα διορθωθεί σε επόμενη έκδοση του συντακτικού αναλυτή.