ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ
ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ (Α ΚΑΙ Β ΜΕΡΟΣ)

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

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Περιεχόμενα

ΜΕΡΟΣ Α-1: Εκμάθηση σύνταξης ΚΕ

Σε αυτό το μέρος της εργασίας (του μέρους Α) μας ζητήθηκε να εξοικειωθούμε και να εξασκηθούμε στην σύνταξη και δημιουργία κανονικών εκφράσεων για την παραγωγή λεξημάτων. Σε αυτό μέρος δεν ζητήθηκε η κατάθεση εργασίας, ο σκοπός του ήταν μόνο η αυτο-εκπαίδευση μας.

ΜΕΡΟΣ Α-2: Κωδικοποίηση αυτομάτων Πεπερασμένων καταστάσεων μέσω FSM

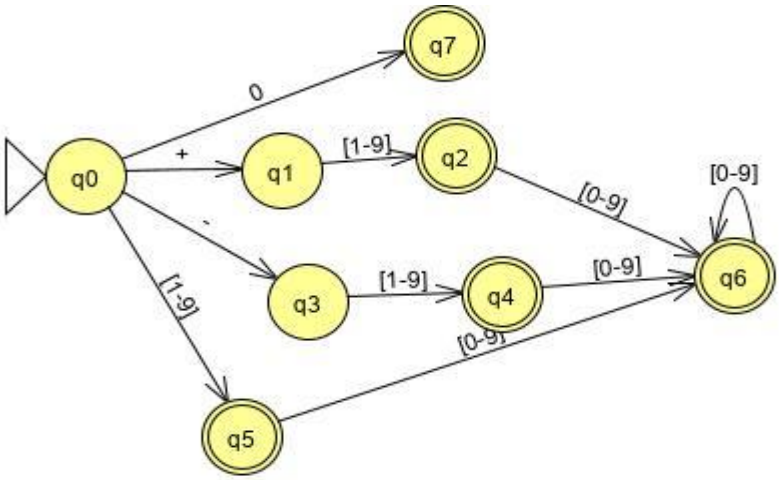
Σε αυτό το μέρος της εργασίας (του μέρους Α) μας ζητήθηκε να σχεδίασουμε σε EBNF τις ΚΕ (κανονικές εκφράσεις) που περιγράφουν τις λεκτικές μονάδες της γλώσσας της εργασίας μας, τα αντίστοιχα πεπερασμένα αυτόματα αναγνώρισής τους και το ενιαίο αυτόματο που προκύπτει από αυτά (παραλείοντας τις κρατημένες λέξεις, όπως π.χ. εντολές της γλώσσας). Επιπλέον ζητήθηκε να προσομοιώσουμε το ενιαίο αυτόματο σε έναν γενικό Πίνακα Μεταβάσεων (ΠΜ) και στη συνέχεια, με τη βοήθεια του μετα-εργαλείου FSM να κωδικοποιήσουμε τον γενικό ΠΜ για να ελέγξουμε την ορθή αναγνώριση των λεκτικών μονάδων της γραμματικής. Παρακάτω ακολουθεί η τεκμηρίωση και η υλοποίηση των κανονικών εκφράσεων για την αναγνώριση των λεκτικών μονάδων, μαζί με τα αυτόματα πεπερασμένων καταστάσεων, τον πίνακα μεταβάσεων του ενιαίου αυτομάτου και ο κώδικας FSM. Τέλος ακολουθούν οι εξαντλητικοί έλεγχοι.

Κανονικές εκφράσεις σε EBNF λεκτικών μονάδων της mini CLIPS

Ακολουθούν οι κανονικές εκφράσεις μαζί με τις εικόνες των αυτόματων

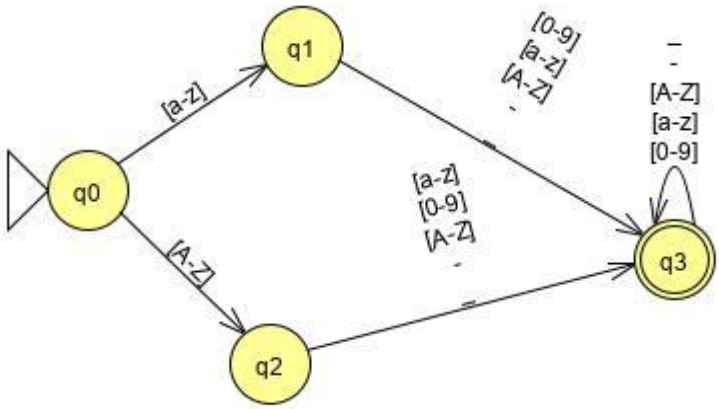
1η κανονική έκφραση: Ακέραιοι αριθμοί

^((\+|-)[1-9]|[1-9]|(^(0)$))[0-9]*$



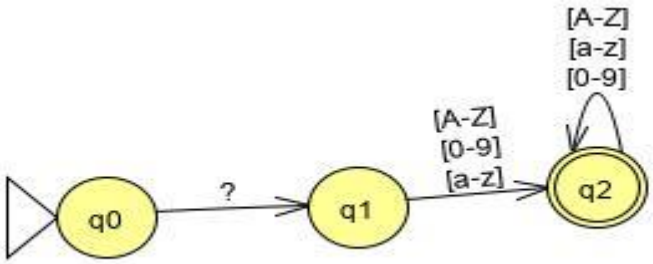
2η κανονική έκφραση: Ονόματα ορισμών και άλλων στοιχειών μέσα σε γεγονότα

^([a-z]|[A-Z])(\d|[a-z]|[A-Z]|[-_])*$



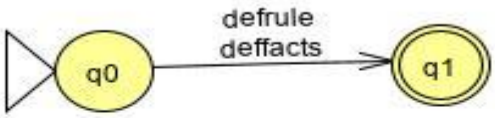
3η κανονική έκφραση: Ονόματα μεταβλητών

^\?(\d|[a-z]|[A-Z])*$



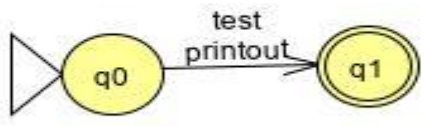
4η κανονική έκφραση: Κρατημένες λέξεις πρωταρχικών συναρτήσεων deffacts, defrule

deffacts|defrule



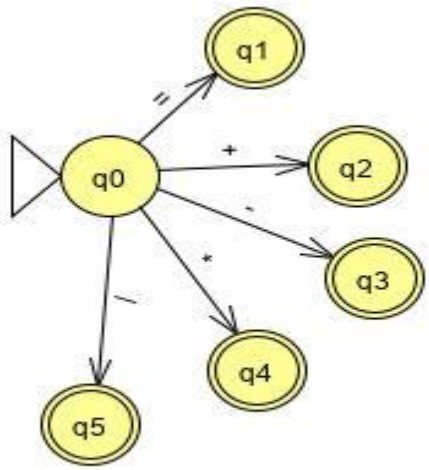
5η κανονική έκφραση: Κρατημένη λέξη συνάρτησης για πραγματοποίηση σύγκρισης test και εκτύπωσης printout

`test|printout`



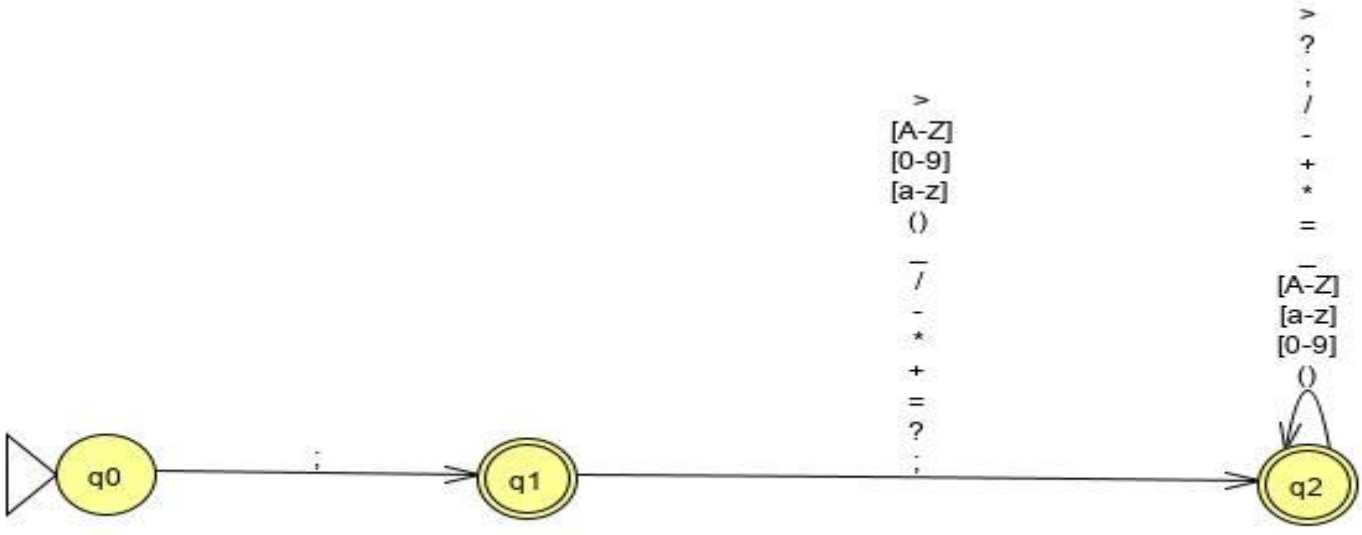
6η κανονική έκφραση: Κρατημένες λέξεις τελεστών =, +, -, *, /

=|\+|-|*|\/



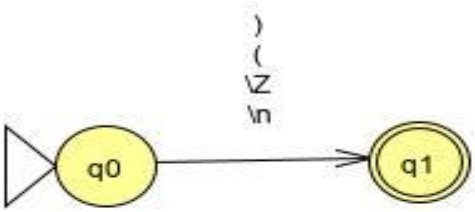
7η κανονική έκφραση: Σχόλια

^;.*$

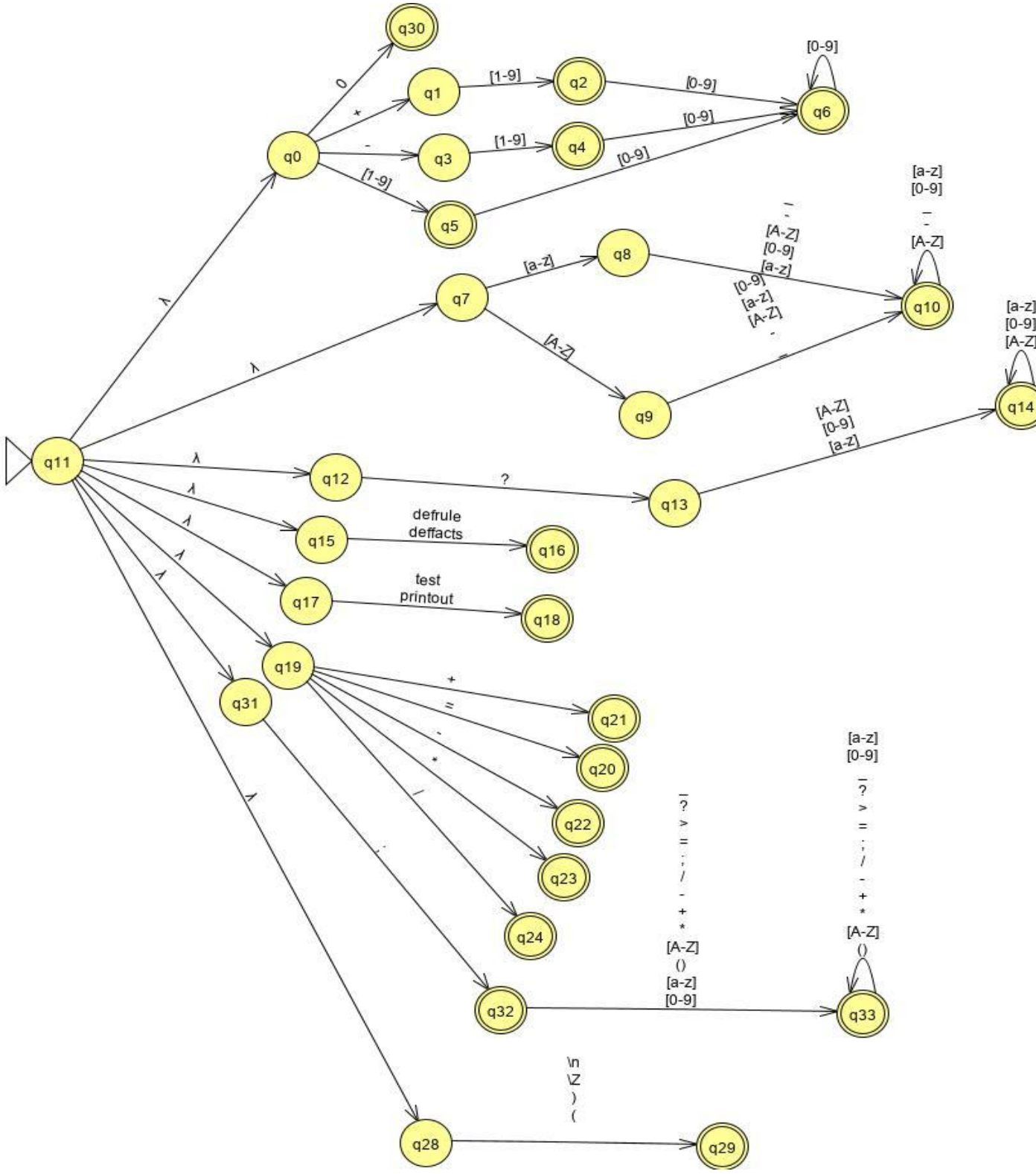


8η κανονική έκφραση: Διαχωριστές

|\n|\Z|\(|\)



ΓΕΝΙΚΟ ΑΥΤΟΜΑΤΟ ΑΝΑΓΝΩΡΙΣΗΣ



ΠΙΝΑΚΑΣ ΜΕΤΑΒΑΣΕΩΝ

	*	/	=	+	-	\s	\n	EOF	(	)	0	0-9	_1-9	a-z	A-Z	;	?	>	_
INIT	FINAL	FINAL	FINAL	TELESTES	TELESTES	FINAL	FINAL	FINAL	FINAL	FINAL	FINAL		AKAIRAIOI_2	ORISMOI	ORISMOI	SXOLIA	METABLITES		
AKAIRAIOI_1							GOOD						AKAIRAIOI_2						
AKAIRAIOI_2							GOOD				AKAIRAIOI_2	AKAIRAIOI_2							
METAVLITES							GOOD				METAVLITES	METAVLITES	METAVLITES	METAVLITES	METAVLITES				
ORISMOI					ORISMOI		GOOD				ORISMOI	ORISMOI	ORISMOI	ORISMOI	ORISMOI				ORISMOI
TELESTES							GOOD						AKAIRAIOI_2						
SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA		GOOD		SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA	SXOLIA
FINAL							GOOD												
GOOD																			

ΚΩΔΙΚΑΣ FSM

```
START=INIT

INIT:

\* -> FINAL
\/ -> FINAL
= -> FINAL
\s -> FINAL
\n -> FINAL
EOF -> FINAL
( -> FINAL
) -> FINAL
0 -> FINAL

+ -> TELESTES
- -> TELESTES

1-9 -> AKERAIOI_2

a-z -> ORISMOI
A-Z -> ORISMOI

; -> SXOLIA

? -> METAVLITES

AKERAIOI_1:
1-9 -> AKERAIOI_2
\n -> GOOD

AKERAIOI_2:
0-9 -> AKERAIOI_2
\n -> GOOD

METAVLITES:
0-9 -> METAVLITES
a-z -> METAVLITES
A-Z -> METAVLITES
\n -> GOOD

ORISMOI:
```

```
a-z -> ORISMOI
A-Z -> ORISMOI
0-9 -> ORISMOI
- -> ORISMOI
_ -> ORISMOI
\n -> GOOD

TELESTES:
1-9 -> AKERAIOI_2
\n -> GOOD

SXOLIA:
0-9 -> SXOLIA
a-z -> SXOLIA
A-Z -> SXOLIA
; -> SXOLIA
? -> SXOLIA
+ -> SXOLIA
- -> SXOLIA
() -> SXOLIA
* -> SXOLIA
/ -> SXOLIA
= -> SXOLIA
> -> SXOLIA
_ -> SXOLIA
\n -> GOOD

FINAL:
\n -> GOOD

GOOD(OK):
```

<u>ΣΗΜΕΙΩΣΗ</u>:

Μετάβαση AKERIAIOI_1 & AKERAIOI_2. Οι ακέραιοι χωρίζονται σε δύο διαφορετικές μεταβάσεις λόγω του μηδενός και των πρόσημων. Λόγω της ιδιαιτερότητας του μηδενός, δηλαδή το γεγονός πως δεν έχει πρόσημο, έπρεπε να διαχωριστεί ο κώδικας σε δύο παρακλάδια. Στην πρώτη περίπτωση (Μετάβαση AKERAIOI_1) , αν δοθεί ως όρισμα ένας εκ των τελεστών «+» ή «-» τότε ο αριθμός που θα ακολουθεί ΔΕΝ μπορεί να είναι ο «0». Μπορεί να είναι όμως οποιοσδήποτε εκ των 1-9, άρα αυτοί θα είναι μέσα στην Μετάβαση AKERIAIOI_1 μαζί με τον χαρακτήρα new_line (\n) φυσικά. Στην Μετάβαση AKERIAIOI_2 υπάρχουν όλοι οι αριθμοί διαθέσιμοι μαζί με τον χαρακτήρα new_line (\n).

ΕΞΑΝΤΛΗΤΙΚΟΙ ΕΛΕΓΧΟΙ ΚΩΔΙΚΑ

ΕΛΕΓΧΟΣ ΑΚΕΡΑΙΩΝ

ΟΡΙΣΜΑ: 0

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:
ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ "0" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΝ ΑΡΙΘΜΟ "0".

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
0
init 0 -> final
final \n -> good
^Z
[14]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΓΙΑΤΙ ΤΟ ΜΗΔΕΝ ΟΡΙΖΕΤΑΙ ΩΣ ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΚΕΡΑΙΟΣ ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: +0

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ "0" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΝ ΑΡΙΘΜΟ "0".

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
+0
```

```
init + -> akeraioi_1
fsm: in META_1.fsm, state 'akeraioi_1' input 0 not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΓΙΑΤΙ ΤΟ ΜΗΔΕΝ ΟΡΙΖΕΤΑΙ ΩΣ ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΚΕΡΑΙΟΣ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΟΡΙΣΜΑ: -0

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ "0" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΝ ΑΡΙΘΜΟ "0".

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
-0
init + -> akeraioi_1
fsm: in META_1.fsm, state 'akeraioi_1' input 0 not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΓΙΑΤΙ ΤΟ ΜΗΔΕΝ ΟΡΙΖΕΤΑΙ ΩΣ ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΚΕΡΑΙΟΣ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΟΡΙΣΜΑ: ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΡΙΘΜΟΣ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΕΝΑΝ ΤΥΧΑΙΟ, ΧΩΡΙΣ ΠΡΟΣΗΜΟ, ΑΡΙΘΜΟ ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΥΣ ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ ΑΡΙΘΜΟΥΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
123456789
init 1 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
akeraioi_2 3 -> akeraioi_2
akeraioi_2 4 -> akeraioi_2
akeraioi_2 5 -> akeraioi_2
akeraioi_2 6 -> akeraioi_2
akeraioi_2 7 -> akeraioi_2
```

```
akeraioi_2 8 -> akeraioi_2
akeraioi_2 9 -> akeraioi_2
akeraioi_2 \n -> good
^Z
[16]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ ΩΣ ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: ΑΡΝΗΤΙΚΑ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΡΙΘΜΟΣ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΕΝΑΝ ΤΥΧΑΙΟ, ΜΕ ΑΡΝΗΤΙΚΟ ΠΡΟΣΗΜΟ, ΑΡΙΘΜΟ ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΥΣ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ ΑΡΙΘΜΟΥΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
-1023456789
init - -> telestes
akeraioi_1 1 -> akeraioi_2
akeraioi_1 0 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
akeraioi_2 3 -> akeraioi_2
akeraioi_2 4 -> akeraioi_2
akeraioi_2 5 -> akeraioi_2
akeraioi_2 6 -> akeraioi_2
akeraioi_2 7 -> akeraioi_2
akeraioi_2 8 -> akeraioi_2
akeraioi_2 9 -> akeraioi_2
akeraioi_2 \n -> good
^Z
[18]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ ΩΣ ΑΡΝΗΤΙΚΑ ΠΡΟΣΗΜΑΣΜΕΝΟ,ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: ΘΕΤΙΚΑ ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΡΙΘΜΟΣ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΕΝΑΝ ΤΥΧΑΙΟ, ΜΕ ΘΕΤΙΚΟ ΠΡΟΣΗΜΟ, ΑΡΙΘΜΟ ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ ΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΥΣ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ ΑΡΙΘΜΟΥΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
+1023456789
init + -> telestes
akeraioi_1 1 -> akeraioi_2
akeraioi_1 0 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
akeraioi_2 3 -> akeraioi_2
akeraioi_2 4 -> akeraioi_2
akeraioi_2 5 -> akeraioi_2
akeraioi_2 6 -> akeraioi_2
akeraioi_2 7 -> akeraioi_2
akeraioi_2 8 -> akeraioi_2
akeraioi_2 9 -> akeraioi_2
akeraioi_2 \n -> good
^Z
[18]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ ΩΣ ΘΕΤΙΚΑ ΠΡΟΣΗΜΑΣΜΕΝΟ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑΤΑ: ΛΑΝΘΑΣΜΕΝΑ ΟΡΙΣΜΑΤΑ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑΤΑ ΤΥΧΑΙΕΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
+12as0985gjhkjgfASDFHJGKLHDF2134563
init + -> telestes
akeraioi_1 1 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
fsm: in META_1.fsm, state 'akeraioi_2' input a not accepted

-121345(*&^%$^&*()5676896*&%$
```

```
init - -> telestes
akeraioi_1 1 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
akeraioi_2 1 -> akeraioi_2
akeraioi_2 3 -> akeraioi_2
akeraioi_2 4 -> akeraioi_2
akeraioi_2 5 -> akeraioi_2
fsm: in META_1.fsm, state 'akeraioi_2' input ( not accepted

8765432asdfjhgkhg(*&#@!#$%^()_dfszghjkj1234563789
init 8 -> akeraioi_2
akeraioi_2 7 -> akeraioi_2
akeraioi_2 6 -> akeraioi_2
akeraioi_2 5 -> akeraioi_2
akeraioi_2 4 -> akeraioi_2
akeraioi_2 3 -> akeraioi_2
akeraioi_2 2 -> akeraioi_2
fsm: in META_1.fsm, state 'akeraioi_2' input a not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΑ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΙΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΩΣ ΜΗ ΑΚΕΡΑΙΟΥΣ ΑΡΙΘΜΟΥΣ, ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΕΛΕΓΧΟΣ ΟΡΙΣΜΩΝ

ΟΡΙΣΜΑ: mA98-78_nQk

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ "mA98-78_nQk" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΟΡΙΣΜΟΥΣΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΥΣ ΜΙΚΡΟΥΣ ΛΑΤΙΝΙΚΟΥΣ ΧΑΡΑΚΤΗΡΕΣ, ΑΚΟΛΟΥΘΟΥΜΕΝΟΙ ΜΕ ΟΤΙ ΣΥΜΒΟΛΟ, ΑΡΙΘΜΟ ΚΑΙ ΓΡΑΜΜΑ ΜΠΟΡΟΥΝ ΝΑ ΔΕΧΤΟΥΝ ΩΣ ΟΡΙΣΜΑ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
mA98-78_nQk
init m -> orismoi
orismoi A -> orismoi
orismoi 9 -> orismoi
orismoi 8 -> orismoi
orismoi - -> orismoi
orismoi 7 -> orismoi
orismoi 8 -> orismoi
```

```
orismoi _ -> orismoi
orismoi n -> orismoi
orismoi Q -> orismoi
orismoi k -> orismoi
orismoi \n -> good
^Z
[19]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΓΙΑΤΙ Ο ΚΩΔΙΚΑΣ ΑΝΑΓΝΩΡΙΣΕ ΟΛΟΚΛΗΡΟ ΤΟ ΟΡΙΣΜΑ ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: MA98-78_nQk

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ "mA98-78_nQk" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΟΡΙΣΜΟΥΣΚΑΙ ΕΙΔΙΚΟΤΕΡΑ ΓΙΑ ΤΟΥΣ ΜΙΚΡΟΥΣ ΛΑΤΙΝΙΚΟΥΣ ΧΑΡΑΚΤΗΡΕΣ, ΑΚΟΛΟΥΘΟΥΜΕΝΟΙ ΜΕ ΟΤΙ ΣΥΜΒΟΛΟ, ΑΡΙΘΜΟ ΚΑΙ ΓΡΑΜΜΑ ΜΠΟΡΟΥΝ ΝΑ ΔΕΧΤΟΥΝ ΩΣ ΟΡΙΣΜΑ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
MA98-78_nQk
init M -> orismoi
orismoi A -> orismoi
orismoi 9 -> orismoi
orismoi 8 -> orismoi
orismoi - -> orismoi
orismoi 7 -> orismoi
orismoi 8 -> orismoi
orismoi _ -> orismoi
orismoi n -> orismoi
orismoi Q -> orismoi
orismoi k -> orismoi
orismoi \n -> good
^Z
[19]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΓΙΑΤΙ Ο ΚΩΔΙΚΑΣ ΑΝΑΓΝΩΡΙΣΕ ΟΛΟΚΛΗΡΟ ΤΟ ΟΡΙΣΜΑ ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑΤΑ: ΛΑΝΘΑΣΜΕΝΑ ΟΡΙΣΜΑΤΑ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΩΝ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΜΙΑ ΤΥΧΑΙΑ ΣΥΜΒΟΛΟΣΕΙΡΑ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΟΡΙΣΜΟΥΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
-aAQ+)*z
init - -> akeraioi_1
fsm: in META_1.fsm, state 'akeraioi_1' input a not accepted

_aAQ+)*z
fsm: in META_1.fsm, state 'init' input _ not accepted

aA*&yt
init a -> orismoi
orismoi A -> orismoi
fsm: in META_1.fsm, state 'orismoi' input * not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΑ ΕΠΕΙΔΗ ΔΕΝ ΑΝΑΓΝΩΡΙΣΕ ΤΙΣ ΣΥΜΒΟΛΟΣΕΙΡΑΕΣ ΩΣ ΟΡΙΣΜΑΤΑ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΣΗΜΕΙΩΣΗ: ΣΤΑ ΠΡΩΤΑ ΔΥΟ ΤΡΕΞΙΜΑΤΑ ΕΔΩΣΑ ΤΑ ΣΥΜΒΟΛΑ ΤΑ ΟΠΟΙΑ ΕΠΙΤΡΕΠΟΝΤΑΙ ΣΤΑ ΟΡΙΣΜΑΤΑ, ΟΜΩΣ ΕΝΑ ΟΡΙΣΜΑ ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΞΕΚΙΝΑ ΜΕ ΑΥΤΑ. ΟΠΟΤΕ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΗΤΑΝ ΟΡΘΟ.

ΕΛΕΓΧΟΣ ΜΕΤΑΒΛΗΤΩΝ

ΟΡΙΣΜΑ: ?a89zB3

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "?" ΑΚΟΛΟΥΘΟΥΜΕΝΟ ΑΠΟ ΜΙΑ ΤΥΧΑΙΑ ΣΥΜΒΟΛΟΣΕΙΡΑ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
?a89zB3
init ? -> metavlites
metavlites a -> metavlites
```

```
metavlites 8 -> metavlites
metavlites 9 -> metavlites
metavlites z -> metavlites
metavlites B -> metavlites
metavlites 3 -> metavlites
metavlites \n -> good
^Z
[20]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΗΝ ΣΥΜΒΟΛΟΣΕΙΡΑ ΩΣ ΜΕΤΑΒΛΗΤΗ ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑΤΑ: ΛΑΝΘΑΣΜΕΝΑ ΟΡΙΣΜΑΤΑ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "?" ΑΚΟΛΟΥΘΟΥΜΕΝΟ ΑΠΟ ΜΙΑ ΤΥΧΑΙΑ ΣΥΜΒΟΛΟΣΕΙΡΑ, Η ΟΠΟΙΑ ΠΕΡΙΕΧΕΙ ΣΥΜΒΟΛΑ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
?A=>B-G+D^90
init ? -> metavlites
metavlites A -> metavlites
fsm: in META_1.fsm, state 'metavlites' input = not accepted

?z=>B-G+D^90
init ? -> metavlites
metavlites z -> metavlites
fsm: in META_1.fsm, state 'metavlites' input = not accepted

?5=>B-G+D^90
init ? -> metavlites
metavlites 5 -> metavlites
fsm: in META_1.fsm, state 'metavlites' input = not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΑ ΕΠΕΙΔΗ ΔΕΝ ΑΝΑΓΝΩΡΙΣΕ ΤΙΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΩΣ ΜΕΤΑΒΛΗΤΕΣ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΟΡΙΣΜΑ: =

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "=" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
=
init = -> final
final \n -> good
^Z
[21]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "=" ΩΣ ΤΕΛΕΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: +

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "+" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
init + -> telestes
telestes \n -> good
^Z
[22]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "+" ΩΣ ΤΕΛΕΣΤΗ,ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: -

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "-" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
-
init - -> telestes
telestes \n -> good
^Z
[22]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "-" ΩΣ ΤΕΛΕΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: *

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "*" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
*
init * -> final
final \n -> good
^Z
[21]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "*" ΩΣ ΤΕΛΕΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: /

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "/" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
/
```

```
init / -> final
final \n -> good
^Z
[21]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "/" ΩΣ ΤΕΛΕΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑΤΑ: ΛΑΝΘΑΣΜΕΝΑ ΟΡΙΣΜΑΤΑ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑΤΑ ΤΟ ΣΥΜΒΟΛΑ ΤΩΝ ΤΕΛΕΣΤΩΝ ΑΚΟΛΟΥΘΟΥΜΕΝΑ ΑΠΟ ΤΥΧΑΙΕΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
=3EP
init = -> final
fsm: in META_1.fsm, state 'final' input 3 not accepted

-a67z
init - -> telestes
fsm: in META_1.fsm, state 'telestes' input a not accepted

+A45y
init + -> telestes
fsm: in META_1.fsm, state 'telestes' input A not accepted

*h56#^((*
init * -> final
fsm: in META_1.fsm, state 'final' input h not accepted

/x67)(*&
init / -> final
fsm: in META_1.fsm, state 'final' input x not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΑ ΕΠΕΙΔΗ ΔΕΝ ΑΝΑΓΝΩΡΙΣΕ ΤΙΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΩΣ ΤΕΛΕΣΤΕΣ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΕΛΕΓΧΟΣ ΣΧΟΛΙΩΝ

ΟΡΙΣΜΑ: ;asgjd0985

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ ";" ΑΚΟΛΟΥΘΟΥΜΕΝΟ ΑΠΟ ΜΙΑ ΤΥΧΑΙΑ ΣΥΜΒΟΛΟΣΕΙΡΑ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
;agjd0985?/
init ; -> sxolia
sxolia a -> sxolia
sxolia s -> sxolia
sxolia g -> sxolia
sxolia j -> sxolia
sxolia d -> sxolia
sxolia 0 -> sxolia
sxolia 9 -> sxolia
sxolia 8 -> sxolia
sxolia 5 -> sxolia
sxolia ? -> sxolia
sxolia / -> sxolia
sxolia \n -> good
^Z
[23]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΗΝ ΣΥΜΒΟΛΟΣΕΙΡΑΣ ΣΧΟΛΙΟ ΛΟΓΟ ΤΟΥ ΣΥΜΒΟΛΟΥ ";" ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΣΗΜΕΙΩΣΗ: ΔΕΝ ΜΠΟΡΩ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΟΡΘΟΤΗΤΑ ΤΟΥ ΚΩΔΙΚΑ ΤΩΝ ΣΧΟΛΙΩΝ ΜΕ ΛΑΝΘΑΣΜΕΝΗ ΕΙΣΟΔΟ ΓΙΑ ΤΟΝ ΛΟΓΟ ΟΤΙ ΠΙΣΩ ΑΠΟ ΤΟ ΣΥΜΒΟΛΟ ";", ΠΟΥ ΥΠΟΔΗΛΩΝΕΙ ΤΗΝ ΑΡΧΗ ΣΧΟΛΙΟΥ, ΜΠΟΡΕΙ ΝΑ ΜΠΕΙ ΟΠΟΙΑΔΗΠΟΤΕ ΣΥΜΒΟΛΟΣΕΙΡΑ, ΑΝΕΞΑΡΤΗΤΩΣ ΑΡΙΘΜΟΥ,ΓΡΑΜΜΑΤΟΣ Η ΣΥΜΒΟΛΟΥ, ΕΝΤΟΣ ΤΩΝ ΟΡΙΩΝ ΤΗΣ ΓΛΩΣΣΑΣ MINI-CLIPS. ΑΡΑ ΕΙΤΕ ΔΕΝ ΘΑ ΒΑΛΩ ΤΟ ΣΥΜΒΟΛΟ ";", Η ΘΑ ΠΡΕΠΕΙ ΝΑ ΞΕΦΥΓΩ ΕΝΤΟΣ ΟΡΙΩΝ ΤΗΣ ΓΛΩΣΣΑΣ MINI-CLIPS ΩΣΤΕ ΝΑ ΔΩΣΩ ΑΠΟΤΕΛΕΣΜΑ ΜΕ ΛΑΝΘΑΣΜΕΝΗ ΕΙΣΟΔΟ ΚΩΔΙΚΑ.

ΕΛΕΓΧΟΣ ΔΙΑΧΩΡΙΣΤΩΝ

ΟΡΙΣΜΑ:ΚΕΝΟ (SPACE)

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "/" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
init \s -> final
final \n -> good
^Z
[24]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ ΚΕΝΟ ΩΣ ΔΙΑΧΩΡΙΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: (

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "/" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
(
init ( -> final
final \n -> good
^Z
[26]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ "(" ΩΣ ΔΙΑΧΩΡΙΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ:)

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "/" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
)
init ) -> final
final \n -> good
^Z
[26]+  σταματημένο  ./fsm -trace META_1
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ ")" ΩΣ ΔΙΑΧΩΡΙΣΤΗ,ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑ: ENTER (ΑΛΛΑΓΗ ΓΡΑΜΜΗΣ)

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑ ΤΟ ΣΥΜΒΟΛΟ "/" ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΙΣ ΜΕΤΑΒΛΗΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
init \n -> final
^Z
[27]+  σταματημένο  ./fsm -trace META_1
```

ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΟ ΕΠΕΙΔΗ ΑΝΑΓΝΩΡΙΣΕ ΤΟ ENTER ΩΣ ΔΙΑΧΩΡΙΣΤΗ, ΟΠΟΤΕ Ο ΚΩΔΙΚΑΣ ΜΠΗΚΕ ΣΤΟ ΚΑΤΑΛΛΗΛΟ STATE ΚΑΙ ΕΜΦΑΝΙΣΕ ΣΩΣΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ.

ΟΡΙΣΜΑΤΑ: ΛΑΝΘΑΣΜΕΝΑ ΟΡΙΣΜΑΤΑ

ΕΠΕΞΗΓΗΣΗ ΟΡΙΣΜΑΤΟΣ:

ΔΙΝΩ ΩΣ ΟΡΙΣΜΑΤΑ ΤΟΥΣ ΔΙΑΧΩΡΙΣΤΕΣ ΑΚΟΛΟΥΘΟΥΜΕΝΑ ΑΠΟ ΤΥΧΑΙΕΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ, ΩΣΤΕ ΝΑ ΕΛΕΓΞΩ ΤΗΝ ΣΥΝΤΑΞΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΓΙΑ ΤΟΥΣ ΔΙΑΧΩΡΙΣΤΕΣ.

ΑΠΟΤΕΛΕΣΜΑ ΚΩΔΙΚΑ:

```
init \n -> final
23246
fsm: in META_1.fsm, state 'final' input 2 not accepted
```

```
init \s -> final
final \n -> good
09-876LKNBJHVCG
fsm: in META_1.fsm, state 'good' input 0 not accepted

(0986)
init ( -> final
fsm: in META_1.fsm, state 'final' input 0 not accepted

)(#$%
init ) -> final
fsm: in META_1.fsm, state 'final' input ( not accepted
```

ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΚΩΔΙΚΑ:

ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΚΩΔΙΚΑ ΕΙΝΑΙ ΟΡΘΑ ΕΠΕΙΔΗ ΔΕΝ ΑΝΑΓΝΩΡΙΣΕ ΤΙΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΩΣ ΔΙΑΧΩΡΙΣΤΕΣ ΟΠΟΤΕ ΘΑ ΕΠΡΕΠΕ ΝΑ ΜΟΥ ΕΠΙΣΤΡΑΦΕΙ ΜΗ ΟΡΘΗ ΣΥΝΤΑΞΗ, ΟΠΩΣ ΚΑΙ ΕΓΙΝΕ.

ΜΕΡΟΣ Α-3: συμπλήρωση πρότυπου κώδικα flex

Σε αυτό το μέρος της εργασίας (του μέρους Α) μας ζητήθηκε να ολοκληρώσετε τον πρότυπο ημιτελή κώδικα simple-flex-code.l μέσα από το συμπιεσμένο αρχείο που υπάρχει στο eclass συμπληρώνοντας τα “FILL ME” στον κώδικα Flex και στο αρχείο Token.h και στην συνέχεια μετά την δοκιμή του κώδικα να ολοκληρώσουμε τον λεκτικό μας αναλυτή όλα τα στοιχεία της γλώσσας mini-CLIPS (λεκτικές μονάδες, διαχωριστές και σχόλια). Παρακάτω ακολουθεί η υλοποίηση των εξαντητικών ελέγχων για τον έλεγχο της ορθής λειτουργίας του λεκτικού αναλυτή.

ΕΞΑΝΤΗΤΙΚΟΙ ΕΛΕΓΧΟΙ ΚΩΔΙΚΑ

Ελεγχος ακεραίων

Όρισμα 0: Δίνω ως όρισμα το «0» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «0».

Αποτέλεσμα στο output.txt:

Line=1,token=INTCONST, value=”0”

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το μηδέν ορίζεται ως μη προσημασμένος ακέραιος οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα +0: Δίνω ως όρισμα το «+0» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «+0».

Αποτέλεσμα στο output.txt:

Line=2, token=UNKNOWN TOKEN, value="+0"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το μηδέν ορίζεται ως μη προσημασμένος ακέραιος και το «+0» δεν είναι ορθό οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα -0: Δίνω ως όρισμα το «-0» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «-0».

Αποτέλεσμα στο output.txt:

Line=3, token=UNKNOWN TOKEN, value="-0"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το μηδέν ορίζεται ως μη προσημασμένος ακέραιος και το «-0» δεν είναι ορθό οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα 123456789: Δίνω ως όρισμα το «123456789» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «123456789».

Αποτέλεσμα στο output.txt:

Line=4, token=INTCONST, value="123456789"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «123456789» ορίζεται ως μη προσημασμένος ακέραιος οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα -1023456789: Δίνω ως όρισμα το «-1023456789» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «-1023456789».

Αποτέλεσμα στο output.txt:

Line=5, token=INTCONST, value="-1023456789"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «-1023456789» ορίζεται ως αρνητικά προσημασμένος ακέραιος οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα +1023456789: Δίνω ως όρισμα το «+1023456789» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους ακέραιους αριθμούς και ειδικότερα για το «+1023456789».

Αποτέλεσμα στο output.txt:

Line=6, token=INTCONST, value="+1023456789"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «+1023456789» ορίζεται ως θετικά προσημασμένος ακέραιος οπότε ο κώδικας εκτελέστηκε σωστά.

Λανθασμένοι ακέραιοι:

Δόθηκαν συμβολοσειρές προκειμένου να ελεγχθεί η ορθή λειτουργία του αναλυτή.

Αποτέλεσμα στο output.txt:

Line=7, token=UNKNOWN TOKEN, value="+12as0985gjhkjgfASDFHJGKLHDF2134563"
Line=8, token=UNKNOWN TOKEN, value="-121345(*&^%$^&*()5676896*&%$"
Line=9,token=UNKNOWN TOKEN
value="8765432asdfjhgkhg(*&#@!#$%^()_dfszghjkj1234563789"

Όπως φαίνεται από την τιμή του token η οποία είναι UNKNOWN TOKEN το αποτέλεσμα δεν είναι ορθό.

Έλεγχος ορισμών

Όρισμα Man:Δίνω ως όρισμα το «Man» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους κεφαλαίους λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=10, token=ARGUMENT, value="Man"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «Man» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα Kostas6:Δίνω ως όρισμα το «Kostas6» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους κεφαλαίους λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=12, token=ARGUMENT, value="Kostas6"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «Kostas6» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα MA98-78_nQk:Δίνω ως όρισμα το «MA98-78_nQk» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους κεφαλαίους λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=15, token=ARGUMENT, value="MA98-78_nQk"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «MA98-78_nQk» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα mannos:Δίνω ως όρισμα το «mannos» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους μικρούς λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=11, token=ARGUMENT, value="mannos"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «Man» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα costas8:Δίνω ως όρισμα το «costas8» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους μικρούς λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=13, token=ARGUMENT, value="costas8"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «costas8» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Όρισμα mA98-78_nQk:Δίνω ως όρισμα το «mA98-78_nQk» ώστε να ελέγξω την σωστή λειτουργία του λεκτικού αναλυτή για τους μικρούς λατινικούς χαρακτήρες, ακολουθούμενοι με ότι σύμβολο, αριθμό και γράμμα μπορούν να δεχτούν ως όρισμα.

Αποτέλεσμα στο output.txt:

Line=14, token=ARGUMENT, value="mA98-78_nQk"

Το αποτέλεσμα του κώδικα είναι ορθό γιατί το «mA98-78_nQk» ακολουθεί τους κανόνες ορισμών της mini-CLIPS οπότε ο κώδικας εκτελέστηκε σωστά.

Λανθασμένοι ορισμοί:

Δόθηκαν συμβολοσειρές προκειμένου να ελεγχθεί η ορθή λειτουργία του αναλυτή.

Αποτέλεσμα στο output.txt:

Line=16, token=UNKNOWN TOKEN, value="-aAQ+)*z"
Line=17, token=UNKNOWN TOKEN, value="_aAQ+)*z"
Line=18, token=UNKNOWN TOKEN, value="aA*&yt"

Όπως φαίνεται από την τιμή του token η οποία είναι UNKNOWN TOKEN το αποτέλεσμα δεν είναι ορθό.

Έλεγχος μεταβλητών

Ορίσματα ?a89zB3,?63,?5A: Δίνω ως όρισμα το σύμβολο «?» ακολουθούμενο από μια τυχαία
συμβολοσειρά με γράμματα και αριθμούς, ώστε να ελέγξω την την σωστή λειτουργία του λεκτικού αναλυτή.

Αποτέλεσμα στο output.txt:

Line=19, token=VARIABLE, value="?a89zB3"
Line=20, token=VARIABLE, value="?63"
Line=21, token=VARIABLE, value="?5A"

Το αποτέλεσμα είναι ορθό γιατί ο αναλυτής αναγνώρισε την συμβολοσειρά ως μεταβλητή.

Λανθασμένες μεταβλητές:

Δόθηκαν συμβολοσειρές με αριθμούς, γράμματα και σύμβολα προκειμένου να ελεγχθεί η ορθή λειτουργία του αναλυτή. Επιπλέον δόθηκαν τυχαία σύμβολα ακολουθούμενα από τυχαίες συμβολοσειρές.

Αποτέλεσμα στο output.txt:

Line=22, token=UNKNOWN TOKEN, value="?A=>B-G+D^90"
Line=23, token=UNKNOWN TOKEN, value="?z=>B-G+D^90"
Line=24, token=UNKNOWN TOKEN, value="?5=>B-G+D^90"
Line=34, token=UNKNOWN TOKEN, value="=3EP"
Line=35, token=UNKNOWN TOKEN, value="-a67z"
Line=36, token=UNKNOWN TOKEN, value="+A45y"
Line=37, token=UNKNOWN TOKEN, value="*h56#^((*"

Line=38, token=UNKNOWN TOKEN, value="/x67)(*&"

Όπως φαίνεται από την τιμή του token η οποία είναι UNKNOWN TOKEN το αποτέλεσμα δεν είναι ορθό.

Έλεγχος δεσμευμένων λέξεων-συμβόλων της γλώσσας

Δίνω τις δεσμευμένες λέξεις-σύμβολα (τελεστές) της γλώσσας για να ελεγχθεί η ορθή λειτουργία του λεκτικού αναλυτή.

Αποτέλεσμα στο output.txt:

Line=25, token=RESERVED1, value="deffacts"
Line=26, token=RESERVED1, value="defrule"
Line=27, token=RESERVED2, value="test"
Line=28, token=RESERVED2, value="printout"
Line=29, token=RESERVED3, value="="
Line=30, token=RESERVED3, value="+"
Line=31, token=RESERVED3, value="-"
Line=32, token=RESERVED3, value="*"
Line=33, token=RESERVED3, value="/"

Το αποτέλεσματα είναι ορθά γιατί ο αναλυτής αναγνώρισε τις συμβολοσειρές και τα σύμβολα ως «RESERVED» (δευσμευμένα).

Έλεγχος διαχωριστών

Όρισματα (και): Δίνω αυτά τα δύο σύμβολα για να ελεγχθεί η ορθή λειτουργία του λεκτικού αναλυτή για αυτά.

Αποτέλεσμα στο output.txt:

Line=40, token=SEPARATOR, value="("

Line=41, token=SEPARATOR, value=")"

Τα αποτελέσματα είναι ορθά γιατί ο αναλυτής τα αναγνωρίζει ως διαχωριστές.

Το όρισμα white_space το αγνοεί ο λεκτικός αναλυτής.

Λανθασμένοι διαχωριστές:

Δόθηκαν λανθασμένοι διαχωριστές προκειμένου να γίνει έλεγχος της ορθότητας του αναλυτή.

Αποτέλεσμα στο output.txt:

Line=42, token=UNKNOWN TOKEN, value=" 23246"
Line=43, token=UNKNOWN TOKEN, value="(0986)"
Line=44, token=UNKNOWN TOKEN, value=")(#$%"

Όπως φαίνεται από την τιμή του token η οποία είναι UNKNOWN TOKEN το αποτέλεσμα δεν είναι ορθό.

ΜΕΡΟΣ Β-1: Ανάπτυξη κώδικα bison για δημιουργία ανεξάρτητου ΣΑ

Σε αυτό το μέρος της εργασίας (του μέρους Β) μας ζητήθηκε να ολοκληρώσουμε τον πρότυπο ημιτελή κώδικα simple-bison-code.y μέσα από το συμπιεσμένο αρχείο στο eclass και στην συνέχεια έπρεπε να ολοκληρώσουμε τον συντακτικό αναλυτή μας ώστε να αναγνωρίζει όσες

περισσότερες εκφράσεις της γλώσσας mini-CLIPS μπορούσαμε. Παρακάτω ακολουθεί η τεκμηρίωση και υλοποίηση των κανόνων παραγωγής του συντακτικού μας αναλυτή μαζί με τους απαραίτητους εξαντλητικούς ελέγχους για την ορθή του λειτουργία.

ΤΕΚΜΗΡΙΩΣΗ-ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΚΑΝΟΝΩΝ ΠΑΡΑΓΩΓΗΣ ΤΟΥ ΣΥΝΤΑΚΤΙΚΟΥ ΑΝΑΛΥΤΗ

Σε αυτό το μέρος η σύνταξη της γλώσσας mini-CLIPS που υλοποιήθηκε είναι η εξής:

- Αριθμητικές πράξεις και σύγκριση (τελεστές =,+,-,/,*) **:** Σε αυτή την έκδοση του συντακτικού μας αναλυτή υλοποιήσαμε αυτές τις λειτουργίες για δύο ή τρεις αριθμούς ή μεταβλητές ή συνδυασμό τους.

Παρακάτω ακολουθεί ένα τμήμα από τους κανόνες παραγωγής για την υλοποίηση των παραπάνω λειτουργίων:

.

.

.

```
| PARENTHESI1 PLUS INTCONST INTCONST PARENTHESI2  { $$ = $3 + $4; }

| PARENTHESI1 PLUS VARIABLE INTCONST PARENTHESI2  { $$ = $3 + $4; }

| PARENTHESI1 PLUS INTCONST VARIABLE PARENTHESI2  { $$ = $3 + $4; }

| PARENTHESI1 PLUS VARIABLE VARIABLE PARENTHESI2  { $$ = $3 + $4; }

| PARENTHESI1 PLUS INTCONST INTCONST INTCONST PARENTHESI2 { $$ = $3 + $4 +
$5; }

| PARENTHESI1 PLUS INTCONST INTCONST VARIABLE PARENTHESI2 { $$ = $3 + $4 +
$5; }
```

```
| PARENTHESI1 PLUS INTCONST VARIABLE INTCONST PARENTHESI2 { $$ = $3 + $4 +
$5; }

.

.

.

| PARENTHESI1 MINUS INTCONST INTCONST PARENTHESI2 { $$ = $3 - $4; }

| PARENTHESI1 MINUS VARIABLE INTCONST PARENTHESI2 { $$ = $3 - $4; }

| PARENTHESI1 MINUS INTCONST VARIABLE PARENTHESI2 { $$ = $3 - $4; }

| PARENTHESI1 MINUS VARIABLE VARIABLE PARENTHESI2 { $$ = $3 - $4; }

| PARENTHESI1 MINUS INTCONST INTCONST INTCONST PARENTHESI2 { $$ = $3 - $4 -
$5; }

| PARENTHESI1 MINUS INTCONST INTCONST VARIABLE PARENTHESI2 { $$ = $3 - $4 -
$5; }

| PARENTHESI1 MINUS INTCONST VARIABLE INTCONST PARENTHESI2 { $$ = $3 - $4 -
$5; }

.

.

.

| PARENTHESI1 MUL INTCONST INTCONST PARENTHESI2 { $$ = $3 * $4; }

| PARENTHESI1 MUL VARIABLE INTCONST PARENTHESI2 { $$ = $3 * $4; }

| PARENTHESI1 MUL INTCONST VARIABLE PARENTHESI2 { $$ = $3 * $4; }

| PARENTHESI1 MUL VARIABLE VARIABLE PARENTHESI2 { $$ = $3 * $4; }

| PARENTHESI1 MUL INTCONST INTCONST INTCONST PARENTHESI2 { $$ = $3 * $4 *
$5; }

.

.
```

```
.

| PARENTHESI1 EQUAL INTCONST INTCONST PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 EQUAL VARIABLE INTCONST PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 EQUAL INTCONST VARIABLE PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 EQUAL VARIABLE VARIABLE PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 EQUAL INTCONST INTCONST INTCONST PARENTHESI2 { $$ = $3 == $4
== $5; }

.

.

.
```

Σε αυτό το απόσπασμα λύπουν αρκετοί από τους κανόνες δίοτι σε επόμενη έκδοση αυτές οι λειτουργίες έγιναν με πιο σωστή και κομψή υλοποίηση.

- Συνάρτηση test**:** Η υλοποίηση της συνάρτησης έγινε για δύο αριθμού ή μεταβλητές ή συνδυασμό τους.

Παρακάτω ακολουθεί το τμήμα από τους κανόνες παραγωγής για την υλοποίηση της συνάρτησης:

```
.

.

.

| PARENTHESI1 TEST VARIABLE INTCONST PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 TEST VARIABLE VARIABLE PARENTHESI2 { $$ = $3 == $4; }

| PARENTHESI1 TEST PARENTHESI1 EQUAL INTCONST INTCONST PARENTHESI2
PARENTHESI2 { $$ = $2 == $4; }

| PARENTHESI1 TEST PARENTHESI1 EQUAL VARIABLE INTCONST PARENTHESI2
PARENTHESI2 { $$ = $5 == $6; }
```

```
| PARENTHESI1 TEST PARENTHESI1 EQUAL INTCONST VARIABLE PARENTHESI2
PARENTHESI2 { $$ = $5 == $6; }

| PARENTHESI1 TEST PARENTHESI1 EQUAL VARIABLE VARIABLE PARENTHESI2
PARENTHESI2 { $$ = $5 == $6; }
```

.

.

.

- Συναρτήσεις deffacts, defrule**:** Η υλοποίηση τους είναι σε αρχικό στάδιο για απλά ή πολλαπλά arguments.

Παρακάτω ακολουθεί το τμήμα από τους κανόνες παραγωγής για την υλοποίηση των συναρτήσεων:

.

.

.

```
| PARENTHESI1 DEFFACTS ARGUMENTS PARENTHESI2 { $$ = $2; }

| PARENTHESI1 DEFFACTS ARGUMENTS PARENTHESI1 ARGUMENTS PARENTHESI2
PARENTHESI2 { $$ = $2, $3, $4; }

| PARENTHESI1 DEFFACTS ARGUMENTS PARENTHESI1 ARGUMENTS PARENTHESI2
PARENTHESI1 ARGUMENTS PARENTHESI2 PARENTHESI2 { $$ = $2, $3, $4, $8; }

//| PARENTHESI1 DEFFACTS ARGUMENTS PARENTHESI1 ARGUMENTS PARENTHESI2
PARENTHESI1 ARGUMENTS PARENTHESI2  PARENTHESI2 { $$ = $2, $3, $4, $8; }


| PARENTHESI1 DEFRULE ARGUMENTS PARENTHESI1 ARGUMENTS PARENTHESI2
PARENTHESI2 { $$ = $2, $3, $4; }

| PARENTHESI1 DEFRULE ARGUMENTS PARENTHESI1 ARGUMENTS PARENTHESI2
PARENTHESI1 ARGUMENTS PARENTHESI2 PARENTHESI2 { $$ = $2, $3, $4, $8; }
```

```
//|  PARENTHESI1  DEFRULE  ARGUMENTS  PARENTHESI1  ARGUMENTS  PARENTHESI2
PARENTHESI1 ARGUMENTS PARENTHESI2  PARENTHESI2 { $$ = $2; }
```

.

.

.

Οι κανόνες που είναι σε σχόλια δεν είχαν τα αναμενόμενα αποτελέσματα.

- Γίνονται αποδεκτά τα ορίσματα (arguments) ανάμεσα σε παρενθέσεις: Η υλοποίηση τους σε αυτή την έκδοση έγινε για δοκιμαστικούς λόγους.

Παρακάτω ακολουθεί το τμήμα από το κανόνα παραγωγής για την αποδοχή των ορισμάτων ανάμεσα σε παρενθέσεις:

.

.

.

```
| PARENTHESI1 ARGUMENTS PARENTHESI2 { $$ = $2; }
```

.

.

.

- Συναρτήση bind: Λειτουργεί για ακέραιους και μεταβλητές.

Παρακάτω ακολουθεί το τμήμα από τους κανόνες παραγωγής για την υλοποίηση της συνάρτησης (αρκετές περιπτώσεις τέθηκαν σε σχόλια λόγω μη αναμενόμενης λειτουργίας):

.

.

.

```
| PARENTHESI1 BIND VARIABLE INTCONST PARENTHESI2 { $$ = $3 = $4; }

| PARENTHESI1 BIND VARIABLE VARIABLE PARENTHESI2 { $$ = $3 = $4; }
```

```
/*

       OI KANONES AYTOI DEN LEITOURGOUN SWSTA EPEIDI O LEKTIKOS ANALYTHS PROSPATHEI NA VREI
TOKEN "()" TO OPOIO DEN YPARXEI.



| PARENTHESI1 BIND VARIABLE PARENTHESI1 READ INTCONST PARENTHESI2 PARENTHESI2 { $$ = $3 = $6;
}

| PARENTHESI1 BIND VARIABLE PARENTHESI1 PLUS INTCONST INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 + $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MINUS INTCONST INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 - $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MUL INTCONST INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 * $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 DIV INTCONST INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 / $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 PLUS INTCONST VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 + $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MINUS INTCONST VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 - $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MUL INTCONST VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 * $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 DIV INTCONST VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 / $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 PLUS VARIABLE INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 + $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MINUS VARIABLE INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 - $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MUL VARIABLE INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 * $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 DIV VARIABLE INTCONST PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 / $7; }
```

```
| PARENTHESI1 BIND VARIABLE PARENTHESI1 PLUS VARIABLE VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 + $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MINUS VARIABLE VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 - $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 MUL VARIABLE VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 * $7; }

| PARENTHESI1 BIND VARIABLE PARENTHESI1 DIV VARIABLE VARIABLE PARENTHESI2 PARENTHESI2 { $$ =
$3 = $6 / $7; }

*/

.

.

.
```

- Συναρτήσεις read και printout**:** Η υλοποιησή τους είναι επιτυχής. Η συνάρτηση printout δέχεται σαν όρισμα το κείμενο που είναι για να εμφανιστεί στην οθόνη.

Παρακάτω ακολουθεί το τμήμα από τους κανόνες παραγωγής για την υλοποίηση των συναρτήσεων:

```
.

.

.

      | PARENTHESI1 READ INTCONST PARENTHESI2 { $$ = $3; }


      | PARENTHESI1 PRINTOUT ARGUMENTS PARENTHESI2 { $$ = $3; }

.

.

.
```

- Σχόλια: Τα σχόλια αναγνωρίζονται και αγνούνται από τον συντακτικό αναλυτή.

Παρακάτω ακολουθεί το τμήμα από τον κανόνα παραγωγής για την υλοποίηση των σχολίων:

```
.

.

.

| COMMENTS { $$ = $1; }

|

;
```

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.

ΜΕΡΟΣ Β-2: Σύνδεση κώδικα Flex με κώδικα Bison

Σε αυτό το μέρος της εργασίας (του μέρους Β) μας ζητήθηκε να ολοκληρώσουμε τον κώδικα bison που δημιουργήσαμε στο προηγούμενο μέρος (δηλαδή να προσθέσουμε τις εκφράσεις που έλυπαν), να προσαρμοστούν κατάλληλα τα αρχεία flex και bison ώστε να γίνει σύνδεση τους, για τα λάθη σύνταξης να καλείται η ρουτίνα διαχείρισης λαθών SyntaxError για να αντιμετωπίσει το πρόβλημα με τη μέθοδο του πανικού και ο μετρητής σημαντικών λαθών (fatal errors) αυξάνεται κατά 1, ενώ η ανάλυση συνεχίζεται στην αρχή της επόμενης συμβολοσειράς εισόδου (εντολής) και τέλος να εμφανίζονται πόσες ήταν οι σωστές εκφράσεις και λέξεις και πόσες οι λάθος.

Σύνδεση κώδικα Flex με κώδικα Bison για την δημιουργία συντακτικού αναλυτή

Σε αυτή την έκδοση του συντακτικού αναλυτή (που είναι υλοποιημένος μαζί με τον λεκτικό αναλυτή) υλοποιήθηκαν όλες οι εντολές της mini-CLIPS και οι κανόνες παραγωγής έχουν αλλάξει και λειτουργούν με αναδρομή, ώστε για παράδειγμα σε μια πράξη να μπορούμε να έχουμε όσους ακέραίους θέλουμε. Επιπλέον για κάθε λειτουργία-εντολή έχουν δημιουργηθεί ξεχωριστοί κανόνες που καταλήγουν στον γενικό (program). Ακολουθεί απόσπασμα κώδικα του αναλυτή για τις μαθηματικές εκφράσεις που έχει υλοποιηθεί ορθότερα:

.

.

.

```
expr:

        INTCONST          { $$ = atoi(yytext); }

        | VARIABLE       { $$ = $1; sosleks++; }

      | PLUS expr praxseis1 { $$ = $2 + $3; fprintf(yyout,"\n\tLINE %d:       SUM\n",line); }

       | MINUS expr praxseis1 { $$ = $2 - $3; fprintf(yyout,"\n\tLINE %d: SUB\n",line);}

        | MUL expr praxseis2 { $$ = $2 * $3; fprintf(yyout,"\n\tLINE %d: MUL\n",line);}

        | DIV expr praxseis2 { if( ( $3 == 0 ) || ($2 == 0) ) { printf("divide by zero\n");
lathosekf++; sosekf--; }

                              else $$ = $2 / $3; fprintf(yyout,"\n\tLINE %d: DIV\n",line); }

        ;

/*

KWDIKAS POU ANAGNWRIZEI MATHIMATIKES PRAXSEIS.
```

```
SE PERIPTWSH DIAIRESHS ME TO "0" EMFANIZETAI MINIMA LATHOUS

KAI AYXSANETE O METRITIS TWN LANTHASMENWN EKFRASEWN.

*/

praxseis1:

        expr     { $$ = $1; }

        | praxseis1 praxseis1 { $$ = $1 + $2;}

        ;

/*

KWDIKAS POU EPITREPEI THN EISAGWGH POLLWN ARITHMWN ME ENA SYMBOLO "+" H "-".

*/



praxseis2:

        expr     { $$ = $1; }

        | praxseis2 praxseis2 { $$ = $1 * $2;}

        ;

/*

KWDIKAS POU EPITREPEI THN EISAGWGH POLLWN ARITHMWN ME ENA SYMBOLO "*" H "/".

*/



expr2:

        PARENTHESI1 expr PARENTHESI2 { $$ = $2; }

        | expr { $$ = $1; }
```

;

.

.

.

Αρχικα διαβάζουμε το σύμβολο που δίνει ο χρήστης και προηγείται των αριθμών, ώστε να ξέρουμε τι ενεργεια θέλει να κανει. Επειτα χρησιμοποιούμε αναδρομική τεχνική ώστε να μπορεί ο χρήστης να δώσει παραπάνω από δύο αριθμούς χρησιμοποιόντας το ίδιο σύμβολο πράξης.

ΕΞΑΝΤΛΗΤΙΚΟΣ ΕΛΕΓΧΟΣ ΕΝΤΟΛΩΝ

Ορθές εντολές

Έλεγχος εντολών υπολογισμού πράξεων ακεραίων (δύο ή περισσότερους ακέραιους) :

Εντολή: `(+ +5 -6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 21: Flex -> Matched token: (
      LINE 21: Flex -> Matched token: +
      LINE 21: Flex -> Matched token: +5
      LINE 21: Flex -> Matched token: -6
      LINE 21: Flex -> Matched token: )

     LINE 21: SUM
-1
```

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

Εντολή: `(+ 4 7)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 25: Flex -> Matched token: (

      LINE 25: Flex -> Matched token: +

      LINE 25: Flex -> Matched token: 4

      LINE 25: Flex -> Matched token: 7

      LINE 25: Flex -> Matched token: )


     LINE 25: SUM
```

```
11
```

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

Εντολή: `(+ 5 +6 7 -7)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 24: Flex -> Matched token: (
      LINE 24: Flex -> Matched token: +
      LINE 24: Flex -> Matched token: 5
      LINE 24: Flex -> Matched token: +6
      LINE 24: Flex -> Matched token: 7
      LINE 24: Flex -> Matched token: -7
      LINE 24: Flex -> Matched token: )

     LINE 24: SUM
11
```

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

Εντολή: `(- 5 72 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 28: Flex -> Matched token: (
      LINE 28: Flex -> Matched token: -
      LINE 28: Flex -> Matched token: 5
```

```
      LINE 28: Flex -> Matched token: 72
      LINE 28: Flex -> Matched token: 5
      LINE 28: Flex -> Matched token: )

     LINE 28: SUB
-72
```

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

Εντολή: `(/ 6 3)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 31: Flex -> Matched token: (
      LINE 31: Flex -> Matched token: /
      LINE 31: Flex -> Matched token: 6
      LINE 31: Flex -> Matched token: 3
      LINE 31: Flex -> Matched token: )

     LINE 31: DIV
2
```

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

Εντολή: `(/ -6 2)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 32: Flex -> Matched token: (
      LINE 32: Flex -> Matched token: /
      LINE 32: Flex -> Matched token: -6
      LINE 32: Flex -> Matched token: 2
      LINE 32: Flex -> Matched token: )

     LINE 32: DIV
-3
```

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

Εντολή: `(* 4 2 6 1 -7 23 0)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 33: Flex -> Matched token: (
      LINE 33: Flex -> Matched token: *
      LINE 33: Flex -> Matched token: 4
      LINE 33: Flex -> Matched token: 2
      LINE 33: Flex -> Matched token: 6
      LINE 33: Flex -> Matched token: 1
      LINE 33: Flex -> Matched token: -7
      LINE 33: Flex -> Matched token: 23
      LINE 33: Flex -> Matched token: 0
      LINE 33: Flex -> Matched token: )

     LINE 33: MUL
0
```

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

Εντολή: `(- 4 2 666 1 -7 23 0)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 34: Flex -> Matched token: (
      LINE 34: Flex -> Matched token: -
      LINE 34: Flex -> Matched token: 4
      LINE 34: Flex -> Matched token: 2
      LINE 34: Flex -> Matched token: 666
      LINE 34: Flex -> Matched token: 1
      LINE 34: Flex -> Matched token: -7
      LINE 34: Flex -> Matched token: 23
      LINE 34: Flex -> Matched token: 0
      LINE 34: Flex -> Matched token: )

     LINE 34: SUB
-681
```

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

Έλεγχος εντολών υπολογισμού πράξεων ακεραίων (δύο ή περισσότερους ακέραιους-μεταβλητών) :

Εντολή: `(- +3 -20 ?x)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 22: Flex -> Matched token: (
      LINE 22: Flex -> Matched token: -
      LINE 22: Flex -> Matched token: +3
      LINE 22: Flex -> Matched token: -20
      LINE 22: Flex -> Matched token: ?x
      LINE 22: Flex -> Matched token: )

     LINE 22: SUB
23
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της αφαίρεσης των δύο ακεραίων και της μεταβλητής). Σε αυτή την περίπτωση η μεταβλητή έχει την τιμή 0.

Εντολή: `(+ ?var ?x 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 27: Flex -> Matched token: (
      LINE 27: Flex -> Matched token: +
      LINE 27: Flex -> Matched token: ?var
      LINE 27: Flex -> Matched token: ?x
      LINE 27: Flex -> Matched token: 5
      LINE 27: Flex -> Matched token: )

     LINE 27: SUM
5
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της πρόσθεσης του ακεραίου και των μεταβλητών). Σε αυτή την περίπτωση οι μεταβλητές έχουν την τιμή 0.

Εντολή: `(- ?var 5 1)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 29: Flex -> Matched token: (
      LINE 29: Flex -> Matched token: -
      LINE 29: Flex -> Matched token: ?var
      LINE 29: Flex -> Matched token: 5
      LINE 29: Flex -> Matched token: 1
      LINE 29: Flex -> Matched token: )

     LINE 29: SUB
-6
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της αφαίρεσης των δύο ακεραίων και της μεταβλητής). Σε αυτή την περίπτωση η μεταβλητή έχει την τιμή 0.

Εντολή: `(/ ?x 2 1)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 30: Flex -> Matched token: (
      LINE 30: Flex -> Matched token: /
      LINE 30: Flex -> Matched token: ?x
      LINE 30: Flex -> Matched token: 2
      LINE 30: Flex -> Matched token: 1
      LINE 30: Flex -> Matched token: )

     LINE 30: DIV
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της διαίρεσης των δύο ακεραίων και της μεταβλητής). Σε αυτή την περίπτωση η μεταβλητή έχει την τιμή 0 και για αυτό τυπώνεται στην οθόνη ότι έγινε η πράξη αυτή με το 0 .

Έλεγχος εντολών υπολογισμού πράξεων μεταβλητών (δύο ή περισσότερες μεταβλητές) :

Εντολή: `(* ?num ?x)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
      LINE 23: Flex -> Matched token: (
      LINE 23: Flex -> Matched token: *
      LINE 23: Flex -> Matched token: ?num
      LINE 23: Flex -> Matched token: ?x
      LINE 23: Flex -> Matched token: )

     LINE 23: MUL
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ του πολλαπλασιασμού των δύο μεταβλητών). Σε αυτή την περίπτωση οι μεταβλητές έχουν την τιμή 0.

Εντολή: `(* ?var ?x)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την πράξη που εκτελέστηκε.

Αποτέλεσμα:

```
LINE 26: Flex -> Matched token: (
      LINE 26: Flex -> Matched token: *
      LINE 26: Flex -> Matched token: ?var
```

```
      LINE 26: Flex -> Matched token: ?x
      LINE 26: Flex -> Matched token: )

     LINE 26: MUL
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ του πολλαπλασιασμού των δύο μεταβλητών). Σε αυτή την περίπτωση οι μεταβλητές έχουν την τιμή 0.

Έλεγχος εντολών σύγκρισης με τον τελεστή = (ακέραιοι) :

Εντολή: `(= 5 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την σύγκριση που εκτελέστηκε (0 ή 1).

Αποτέλεσμα:

```
LINE 16: Flex -> Matched token: (
      LINE 16: Flex -> Matched token: =
      LINE 16: Flex -> Matched token: 5
      LINE 16: Flex -> Matched token: 6
      LINE 16: Flex -> Matched token: )

     LINE 16: EQUAL
0
```

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

Έλεγχος εντολών σύγκρισης με τον τελεστή =(μεταβλητής και ακεραίου) :

Εντολή: `(= ?num 2)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την σύγκριση που εκτελέστηκε (0 ή 1).

Αποτέλεσμα:

```
      LINE 20: Flex -> Matched token: (
      LINE 20: Flex -> Matched token: =
      LINE 20: Flex -> Matched token: ?num
      LINE 20: Flex -> Matched token: 2
      LINE 20: Flex -> Matched token: )

     LINE 20: EQUAL
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της σύγκρισης μεταβλητής-ακεραίου).

Έλεγχος εντολών σύγκρισης με τον τελεστή =(μεταβλητών) **:**

Εντολή: `(= ?var ?x)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την σύγκριση που εκτελέστηκε (0 ή 1).

Αποτέλεσμα:

```
      LINE 17: Flex -> Matched token: (
      LINE 17: Flex -> Matched token: =
      LINE 17: Flex -> Matched token: ?var
      LINE 17: Flex -> Matched token: ?x
      LINE 17: Flex -> Matched token: )

     LINE 17: EQUAL
1
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της σύγκρισης μεταβλητών).

Έλεγχος εντολών σύγκρισης με τον τελεστή =(μεταβλητή ή ακέραιο και αριθμητική πράξη) **:**

Εντολή: `(= (+?num 2) ?x)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την σύγκριση που εκτελέστηκε (0 ή 1).

Αποτέλεσμα:

```
LINE 18: Flex -> Matched token: (
      LINE 18: Flex -> Matched token: =
      LINE 18: Flex -> Matched token: (
      LINE 18: Flex -> Matched token: +
      LINE 18: Flex -> Matched token: ?num
      LINE 18: Flex -> Matched token: 2
      LINE 18: Flex -> Matched token: )

     LINE 18: SUM
      LINE 18: Flex -> Matched token: ?x
      LINE 18: Flex -> Matched token: )

     LINE 18: EQUAL
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της σύγκρισης με μια αριθμητική πράξη και μια μεταβλητή).

Εντολή: `(=100 (+ ?num 2))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί η τιμή από την σύγκριση που εκτελέστηκε (0 ή 1).

Αποτέλεσμα:

```
LINE 19: Flex -> Matched token: (
      LINE 19: Flex -> Matched token: =
      LINE 19: Flex -> Matched token: 100
      LINE 19: Flex -> Matched token: (
      LINE 19: Flex -> Matched token: +
      LINE 19: Flex -> Matched token: ?num
      LINE 19: Flex -> Matched token: 2
      LINE 19: Flex -> Matched token: )

     LINE 19: SUM
      LINE 19: Flex -> Matched token: )

     LINE 19: EQUAL
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της σύγκρισης με μια αριθμητική πράξη και ένα ακέραιο).

Έλεγχος εντολών ανάθεσης τιμών (bind) **:**

Εντολή: `(bind ?var 15)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει η εκχώρηση στην μεταβλητή.

Αποτέλεσμα:

```
(bind ?var 15)
```

```
     Flex -> Matched token: (
     Flex -> Matched token: bind
     Flex -> Matched token: ?var
     Flex -> Matched token: 15
     Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της ανάθεσης ακέραιας τιμής στην μεταβλητή).

Εντολή: `(bind ?num 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει η εκχώρηση στην μεταβλητή.

Αποτέλεσμα:

```
(bind ?num 6)
     Flex -> Matched token: (
     Flex -> Matched token: bind
     Flex -> Matched token: ?num
     Flex -> Matched token: 6
     Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της ανάθεσης ακέραιας τιμής στην μεταβλητή).

Έλεγχος εντολών εμφάνισης κειμένου στην οθόνη (printout t) **:**

Εντολή: `(printout t (This is good))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί στην οθόνη το κείμενο που είναι μέσα στις παρενθέσεις.

Αποτέλεσμα:

```
      LINE 12: Flex -> Matched token: (
      LINE 12: Flex -> Matched token: printout t
      LINE 12: Flex -> Matched token: (
      LINE 12: Flex -> Matched token: This
      LINE 12: Flex -> Matched token: is
      LINE 12: Flex -> Matched token: good
      LINE 12: Flex -> Matched token: )
      LINE 12: Flex -> Matched token: )

     LINE 12: PRINTOUT
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της εμφάνισης κειμένου στην οθόνη).

Εντολή: `(printout t (This-is-a-test))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί στην οθόνη το κείμενο που είναι μέσα στις παρενθέσεις.

Αποτέλεσμα:

```
LINE 13: Flex -> Matched token: (
      LINE 13: Flex -> Matched token: printout t
      LINE 13: Flex -> Matched token: (
      LINE 13: Flex -> Matched token: This-is-a-test
      LINE 13: Flex -> Matched token: )
      LINE 13: Flex -> Matched token: )

     LINE 13: PRINTOUT
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ της εμφάνισης κειμένου στην οθόνη).

Έλεγχος εντολών δημιουργίας απλών γεγονότων **:**

Εντολή: `(Costas)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα απλό γεγονός.

Αποτέλεσμα:

```
LINE 6: Flex -> Matched token: (
      LINE 6: Flex -> Matched token: Costas
      LINE 6: Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός απλού γεγονότος). Αντίστοιχα οι επόμενες εντολές για την δημιουργία απλών γεγονότων έχουν το αναμενόμενο αποτέλεσμα:

Εντολή: `(costas 6)`

Αποτέλεσμα:

```
LINE 7: Flex -> Matched token: (
      LINE 7: Flex -> Matched token: costas
      LINE 7: Flex -> Matched token: 6
      LINE 7: Flex -> Matched token: )
```

Εντολή: `(Costas is_a really good student)`

Αποτέλεσμα:

```
LINE 8: Flex -> Matched token: (
      LINE 8: Flex -> Matched token: Costas
      LINE 8: Flex -> Matched token: is_a
      LINE 8: Flex -> Matched token: really
      LINE 8: Flex -> Matched token: good
      LINE 8: Flex -> Matched token: student
      LINE 8: Flex -> Matched token: )
```

Εντολή: `(His-AM_is_161041)`

Αποτέλεσμα:

```
      LINE 9: Flex -> Matched token: (
      LINE 9: Flex -> Matched token: His-AM_is_161041
```

```
      LINE 9: Flex -> Matched token: )
0
```

Εντολή: `(Impossible is just an opinion)`

Αποτέλεσμα:

```
      LINE 10: Flex -> Matched token: (
      LINE 10: Flex -> Matched token: Impossible
      LINE 10: Flex -> Matched token: is
      LINE 10: Flex -> Matched token: just
      LINE 10: Flex -> Matched token: an
      LINE 10: Flex -> Matched token: opinion
      LINE 10: Flex -> Matched token: )
0
```

Εντολή: `(THE PARSER WORKS)`

Αποτέλεσμα:

```
LINE 11: Flex -> Matched token: (
      LINE 11: Flex -> Matched token: THE
      LINE 11: Flex -> Matched token: PARSER
      LINE 11: Flex -> Matched token: WORKS
      LINE 11: Flex -> Matched token: )
0
```

Έλεγχος εντολών δημιουργίας απλών-πολλαπλών γεγονότων με την εντολή deffacts:

Εντολή: `(deffacts dynamic-facts (packman-at))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα απλό γεγονός.

Αποτέλεσμα:

```
      LINE 4: Flex -> Matched token: (
      LINE 4: Flex -> Matched token: deffacts
      LINE 4: Flex -> Matched token: dynamic-facts
      LINE 4: Flex -> Matched token: (
      LINE 4: Flex -> Matched token: packman-at
      LINE 4: Flex -> Matched token: )
      LINE 4: Flex -> Matched token: )

     LINE 4: DEFFACTS
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός απλού γεγονότος).

Εντολή: `(deffacts dynamic6 (pack))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα απλό γεγονός.

Αποτέλεσμα:

```
      LINE 5: Flex -> Matched token: (
      LINE 5: Flex -> Matched token: deffacts
      LINE 5: Flex -> Matched token: dynamic6
      LINE 5: Flex -> Matched token: (
      LINE 5: Flex -> Matched token: pack
      LINE 5: Flex -> Matched token: )
      LINE 5: Flex -> Matched token: )

     LINE 5: DEFFACTS
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός απλού γεγονότος).

Εντολή: `(deffacts dynamic-facts (packman-at 5 2))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα απλό γεγονός.

Αποτέλεσμα:

```
LINE 3: Flex -> Matched token: (
LINE 3: Flex -> Matched token: deffacts
LINE 3: Flex -> Matched token: dynamic-facts
LINE 3: Flex -> Matched token: (
LINE 3: Flex -> Matched token: packman-at
LINE 3: Flex -> Matched token: 5
LINE 3: Flex -> Matched token: 2
LINE 3: Flex -> Matched token: )
LINE 3: Flex -> Matched token: )

LINE 3: DEFFACTS
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός απλού γεγονότος).

Εντολή: `(deffacts dynamic-facts (packman-at) (hi) (bye))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα πολλαπλό γεγονός.

Αποτέλεσμα:

```
LINE 2: Flex -> Matched token: (
LINE 2: Flex -> Matched token: deffacts
LINE 2: Flex -> Matched token: dynamic-facts
LINE 2: Flex -> Matched token: (
LINE 2: Flex -> Matched token: packman-at
LINE 2: Flex -> Matched token: )
LINE 2: Flex -> Matched token: (
LINE 2: Flex -> Matched token: hi
LINE 2: Flex -> Matched token: )
LINE 2: Flex -> Matched token: (
LINE 2: Flex -> Matched token: bye
LINE 2: Flex -> Matched token: )
LINE 2: Flex -> Matched token: )
```

```
     LINE 2: DEFFACTS
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός πολλαπλού γεγονότος).

Εντολή: `(deffacts static-facts (food-at 4 2) ( food-at 5 2))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένα πολλαπλό γεγονός.

Αποτέλεσμα:

```
      LINE 1: Flex -> Matched token: (
      LINE 1: Flex -> Matched token: deffacts
      LINE 1: Flex -> Matched token: static-facts
      LINE 1: Flex -> Matched token: (
      LINE 1: Flex -> Matched token: food-at
      LINE 1: Flex -> Matched token: 4
      LINE 1: Flex -> Matched token: 2
      LINE 1: Flex -> Matched token: )
      LINE 1: Flex -> Matched token: (
      LINE 1: Flex -> Matched token: food-at
      LINE 1: Flex -> Matched token: 5
      LINE 1: Flex -> Matched token: 2
      LINE 1: Flex -> Matched token: )
      LINE 1: Flex -> Matched token: )

     LINE 1: DEFFACTS
0
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ η δημιουργία ενός πολλαπλού γεγονότος).

Έλεγχος εντολών δημιουργίας κανόνων με την εντολή defrule **:**

Εντολή:

```
(defrule MOVE-UP (packman-at ?x ?y))
(food-at ?z ?y)
->
(printout t(packman has reached food))
```

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένας κανόνας.

Αποτέλεσμα:

```
LINE 35: Flex -> Matched token: (
      LINE 35: Flex -> Matched token: defrule
      LINE 35: Flex -> Matched token: MOVE-UP
      LINE 35: Flex -> Matched token: (
      LINE 35: Flex -> Matched token: packman-at
      LINE 35: Flex -> Matched token: ?x
      LINE 35: Flex -> Matched token: ?y
      LINE 35: Flex -> Matched token: )
      LINE 35: Flex -> Matched token: )

     LINE 35: DEFRULE
      LINE 36: Flex -> Matched token: (
      LINE 36: Flex -> Matched token: food-at
      LINE 36: Flex -> Matched token: ?z
      LINE 36: Flex -> Matched token: ?y
      LINE 36: Flex -> Matched token: )
      LINE 37: Flex -> Matched token: ->
      LINE 38: Flex -> Matched token: (
      LINE 38: Flex -> Matched token: printout t
      LINE 38: Flex -> Matched token: (
      LINE 38: Flex -> Matched token: packman
      LINE 38: Flex -> Matched token: has
      LINE 38: Flex -> Matched token: reached
      LINE 38: Flex -> Matched token: food
      LINE 38: Flex -> Matched token: )
      LINE 38: Flex -> Matched token: )

     LINE 38: PRINTOUT
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ ο κανόνας). Λόγω θεμάτων υλοποίησης της εντολής, η εντολή αναγνωρίζεται σαν τέσσερεις ή περισσότερες ξεχωριστές εντολές.

Εντολή:

```
(defrule MOVE_down (packman-at 5 5))
(food-at 3 ?y)
->
(printout t(packman has not reached food))
```

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να δημιουργηθεί ένας κανόνας.

Αποτέλεσμα:

```
LINE 39: Flex -> Matched token: (

      LINE 39: Flex -> Matched token: defrule

      LINE 39: Flex -> Matched token: MOVE_down

      LINE 39: Flex -> Matched token: (

      LINE 39: Flex -> Matched token: packman-at

      LINE 39: Flex -> Matched token: 5

      LINE 39: Flex -> Matched token: 5

      LINE 39: Flex -> Matched token: )

      LINE 39: Flex -> Matched token: )


     LINE 39: DEFRULE
```

```
 LINE 40: Flex -> Matched token: (

 LINE 40: Flex -> Matched token: food-at

 LINE 40: Flex -> Matched token: 3

 LINE 40: Flex -> Matched token: ?y

 LINE 40: Flex -> Matched token: )

 LINE 41: Flex -> Matched token: ->

 LINE 42: Flex -> Matched token: (

 LINE 42: Flex -> Matched token: printout t

 LINE 42: Flex -> Matched token: (

 LINE 42: Flex -> Matched token: packman

 LINE 42: Flex -> Matched token: has

 LINE 42: Flex -> Matched token: not

 LINE 42: Flex -> Matched token: reached

 LINE 42: Flex -> Matched token: food

 LINE 42: Flex -> Matched token: )

 LINE 42: Flex -> Matched token: )


LINE 42: PRINTOUT
```

Η εκτέλεση της εντολής είναι επιτυχής διότι την αναγνωρίζει ο συντακτικός και το αποτέλεσμα είναι το αναμενόμενο (εδώ ο κανόνας). Λόγω θεμάτων υλοποίησης της εντολής, η εντολή αναγνωρίζεται σαν τέσσερεις ή περισσότερες ξεχωριστές εντολές.

Λανθασμένες εντολές

Έλεγχος εντολών ελέγχου test (ακέραιοι):

Η εντολή test υλοποιήθηκε στον συντακτικό αναλυτή μας αλλά η λειτουργία του δεν είναι ορθή παρόλες τις προσπάθειες μας.

Εντολή: `(test 4 4)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί 0 ή 1 ανάλογα το αποτέλεσμα του ελεγχου.

Αποτέλεσμα:

```
(test 4 4)
     Flex -> Matched token: (
     Flex -> Matched token: test
     Flex -> Matched token: 4
     Flex -> Matched token: 4
     Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι παρόλου που την αναγνωρίζει ο συντακτικός το αποτέλεσμα δεν είναι το αναμενόμενο (εδώ ο έπρεπε να είναι 1).

Εντολή: `(test 7 6)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί 0 ή 1 ανάλογα το αποτέλεσμα του ελεγχου.

Αποτέλεσμα:

```
(test 7 6)
     Flex -> Matched token: (
     Flex -> Matched token: test
     Flex -> Matched token: 7
     Flex -> Matched token: 6
     Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι παρόλου που την αναγνωρίζει ο συντακτικός το αποτέλεσμα δεν είναι το αναμενόμενο (εδώ είναι 0 αλλά αν ήταν ίδιοι οι αριθμοί θα παρέμενε ίδιο το αποτέλεσμα).

Έλεγχος εντολών ελέγχου test (αριθμοί-μεταβλητές) **:**

Εντολή: `(test ?k 5)`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί 0 ή 1 ανάλογα το αποτέλεσμα του ελεγχου.

Αποτέλεσμα:

```
(test ?k 5)
     Flex -> Matched token: (
     Flex -> Matched token: test
     Flex -> Matched token: ?k
     Flex -> Matched token: 5
     Flex -> Matched token: )
0
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι παρόλου που την αναγνωρίζει ο συντακτικός το αποτέλεσμα δεν είναι το αναμενόμενο (εδώ είναι 0 αλλά αν ήταν ίδιοι οι αριθμοί-μεταβλητές θα παρέμενε ίδιο το αποτέλεσμα).

Έλεγχος εντολών ελέγχου test μαζί με τον τελεστή σύγκρισης = **:**

Εντολή: `(test (= 5 6))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί 0 ή 1 ανάλογα το αποτέλεσμα του ελεγχου.

Αποτέλεσμα:

```
(test (= 5 6))
     Flex -> Matched token: (
     Flex -> Matched token: test
     Flex -> Matched token: (
     Flex -> ERROR, line 1 at lexeme '(' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι συντακτικά δεν την αναγνωρίζει ο αναλυτής. (Δεν την υλοποιήσαμε λόγω χρόνου).

Το ίδιο ισχυεί και για τις παρακάτω εντολές:

```
(test (= ?var ?x)
(test (= ?var 4)
(bind ?x (+ 4 5))
(bind ?x (read))
```

Έλεγχος εντολών με τα ποίο πιθανά συντακτικά λάθη:

Εντολή: `+ 5 6`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει πρόσθεη των ακεραίων.

Αποτέλεσμα:

```
+ 5 6
     Flex -> Matched token: +
     Flex -> ERROR, line 1 at lexeme '+' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Λείπουν οι παρενθέσεις).

Εντολή: `- 6 6 4`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει αφαίρεση των ακεραίων.

Αποτέλεσμα:

```
- 6 6 4
     Flex -> Matched token: -
     Flex -> ERROR, line 1 at lexeme '-' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Λείπουν οι παρενθέσεις).

Εντολή: `* 5`

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

Αποτέλεσμα:

```
* 5
     Flex -> Matched token: *
     Flex -> ERROR, line 1 at lexeme '*' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής και δεν γίνεται πράξη πολλαπλασιασμού με ένα ακέραιο. (Λείπουν οι παρενθέσεις).

Εντολή: `5 + 5`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει πρόσθεση των ακεραίων.

Αποτέλεσμα:

```
5 + 5
     Flex -> Matched token: 5
     Flex -> ERROR, line 1 at lexeme '5' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Λείπουν οι παρενθέσεις και είναι λάθος η σειρά που δώθηκαν οι ακέραιοι και ο τελεστής).

Εντολή: `6 / 3;`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει διαίρεση των ακεραίων.

Αποτέλεσμα:

```
6 / 3;
     Flex -> Matched token: 6
     Flex -> ERROR, line 1 at lexeme '6' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Λείπουν οι παρενθέσεις και είναι λάθος να υπάρχει στο τέλος ο χαρακτήρας ;).

Εντολή: `k = 4;`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει ανάθεση του ακεραίου στην μεταβλητή.

Αποτέλεσμα:

```
k = 4;
     Flex -> ERROR, line 1 at lexeme 'k' : Unrecognized token
error!
     Flex -> ERROR, line 1 at lexeme 'k' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Η ανάθεση γίνεται με την εντολή bind μέσα σε παρενθέσεις).

Εντολή: `(print (kati))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει εμφάνιση ενός κειμένου.

Αποτέλεσμα:

```
(print (kati))
     Flex -> Matched token: (
     Flex -> Matched token: print
     Flex -> Matched token: (
     Flex -> ERROR, line 1 at lexeme '(' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής. (Η εμφάνιση κειμένου γίνεται με την εντολή printout t).

Εντολή: `(defrule MOVE_down (packman-at 5 5)) (food-at 3 ?y) -> (printout t(packman has not reached food))`

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να γίνει δημιουργία ενός κανόνα.

Αποτέλεσμα:

```
(defrule MOVE_down (packman-at 5 5)) (food-at 3 ?y) ->
(printout t(packman has not reached food))
     Flex -> Matched token: (
     Flex -> Matched token: defrule
     Flex -> Matched token: MOVE_down
     Flex -> Matched token: (
     Flex -> Matched token: packman-at
     Flex -> Matched token: 5
     Flex -> Matched token: 5
     Flex -> Matched token: )
     Flex -> Matched token: )
     Flex -> Matched token: (
     Flex -> ERROR, line 1 at lexeme '(' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής.

Εντολή: (+)

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Δεν έχει νόημα μια τέτοια εντολή.

Αποτέλεσμα:

```
(+)
     Flex -> Matched token: (
     Flex -> Matched token: +
     Flex -> Matched token: )
     Flex -> ERROR, line 1 at lexeme ')' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής.

Εντολή: (5)

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Δεν έχει νόημα μια τέτοια εντολή.

Αποτέλεσμα:

```
5)
     Flex -> Matched token: (
     Flex -> Matched token: 5
     Flex -> Matched token: )
5
```

Η εκτέλεση της εντολής είναι επιτυχής αλλά δεν έιναι χρήσιμη.

Εντολή: 6

Δίνω την παραπάνω εντολή στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εντολής ως προς το συντακτικό και την εκτέλεση. Δεν έχει νόημα μια τέτοια εντολή.

Αποτέλεσμα:

```
6
     Flex -> Matched token: 6
     Flex -> ERROR, line 1 at lexeme '6' : syntax error
```

Η εκτέλεση της εντολής δεν είναι επιτυχής διότι δεν την αναγνωρίζει ο συντακτικός αναλυτής.

ΜΕΡΟΣ Β-3: Διαχείριση λεκτικών και συντακτικών προειδοποιητικών λαθών

Σε αυτό το μέρος της εργασίας (μέρος Β) ζητήται να γίνει διαχείριση λεκτικών και συντακτικών προειδοποιητικών λαθών. Αυτό σημαίνει ότι απαιτείται ο εμπλουτισμός των κανόνων της γραμματικής και των κανόνων των λεξημάτων. Επιπλέον ζητήται να παραδοθεί η τελική εργασία (δηλαδή όλες οι εργασίες του εξαμήνου με ένα κοινό έγραφο pdf τεκμηρίωσης). Παρακάτω ακολουθεί η τεκμηρίωση και υλοποίηση των κανόνων παραγωγής του συντακτικού μας αναλυτή μαζί με τους απαραίτητους εξαντλητικούς ελέγχους για την ορθή του λειτουργία.

ΤΕΚΜΗΡΙΩΣΗ-ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΚΑΝΟΝΩΝ ΠΑΡΑΓΩΓΗΣ ΤΟΥ ΣΥΝΤΑΚΤΙΚΟΥ ΑΝΑΛΥΤΗ

Σε αυτή την έκδοση του συντακτικού αναλυτή (που είναι υλοποιημένος μαζί με τον λεκτικό αναλυτή) υλοποιήθηκαν όλες οι εντολές της mini-CLIPS και οι κανόνες παραγωγής καθώς και κανονές παραγωγής για την αναγνώριση συχνών λαθών ώστε να εμφανίζονται στην οθόνη κατάλληλα μυνήματα για την διαχείριση τους. (Παράλληλα έγινε προσπάθεια υλοποίσης κανονικών εκφράσεων στον κώδικα flex για την αναγνώριση λανθασμένων λεξημάτων χωρίς επιτυχία.) Επιπλέον για κάθε λειτουργία-εντολή-λανθασμένη εντολή έχουν δημιουργηθεί ξεχωριστοί κανόνες που καταλήγουν στον γενικό (program).

Ακολουθεί απόσπασμα κώδικα (αρχείο bison) του αναλυτή για την αναγνώριση λανθασμένων εντολών.

Ο συντακτικός αναλυτής μας σε αυτή την έκδοση μπορεί να αναγνωρίσει εντολές που δεν έχουν παρενθέσεις, που λείπει μια από τις δύο παρενθέσεις και την περίπτωση που δεν έχει δωθεί ο τελεστής αριθμητικής πράξης εμφανίζοντας κατάλληλα διαμορφωμένο μήνυμα. Στο ακόλουθο απόσπασμα φαίνεται μερικές από τις περιπτώσεις συχνών λαθών σε εντολές με μαθηματικές πράξεις:

.

.

.

```
suxna_lathoi:

        //suxna_lathoi { $$ = $1; }

        | expr praxseis1 { $$ = $1; fprintf(yyout,"\n\tLINE %d: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS\n",line); lathosekf++; }

        | expr praxseis2 { $$ = $1; fprintf(yyout,"\n\tLINE %d: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS\n",line); lathosekf++; }

        | PARENTHESI1 expr praxseis1 PARENTHESI2 { $$ = $1; fprintf(yyout,"\n\tLINE %d: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS\n",line); lathosekf++; }

        | PARENTHESI1 expr praxseis2 PARENTHESI2 { $$ = $1; fprintf(yyout,"\n\tLINE %d: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS\n",line); lathosekf++; }

        | PLUS expr praxseis1 { $$ = $2 + $3; fprintf(yyout,"\n\tLINE %d: ERROR! PROSTHESI XWRIS PARENTHESEIS\n",line); lathosekf++; }

        | MINUS expr praxseis1 { $$ = $2 - $3; fprintf(yyout,"\n\tLINE %d: ERROR! AFAIRESH XWRIS PARENTHESEIS\n",line); lathosekf++; }

.

.

.
```

Επιπλέον είναι δυνατή η αναγνώριση όλων των εντολών της mini-CLIPS όπου λύπουν οι παρενθέσεις. Στο ακόλουθο απόσπασμα φαίνονται μερικές από αυτές τις περιπτώσεις:

.

.

.

```
| DEFFACTS orismata defexpr PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR! DEFFACTS XWRIS ARISTERH PARENTHESI\n",line); lathosekf++; }

      | PARENTHESI1 DEFFACTS orismata defexpr  { fprintf(yyout,"\n\tLINE %d: ERROR! DEFFACTS XWRIS DEXSIA PARENTHESI\n",line); lathosekf++; }

      | PARENTHESI1 DEFRULE defexpr PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR! DEFRULE XWRIS ONOMA\n",line); lathosekf++; }

      | PARENTHESI1 DEFRULE orismata PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR! DEFRULE XWRIS EXPRESSIONS\n",line); lathosekf++; }
```

.

.

.

```
| PARENTHESI1 PRINTOUT gegonota  { fprintf(yyout,"\n\tLINE %d: ERROR! PRINTOUT XWRIS DEKSIA PARENTHESI\n",line); lathosekf++; }
```

.

.

.

```
| TEST eqexpr PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR! TEST XWRIS ARISTERH PARENTHESI\n",line); lathosekf++; }
```

.

.

.

Τέλος είναι δυνατή η αναγνώριση εντολών που δεν έχουν τα κατάλληλα ορίσματα-μεταβλητές για να εκτελεστούν. Για παράδειγμα όταν κάνουμε ανάθεση με την εντολή `bind`

είναι απαραίτητο να έχουμε μια μεταβλητή ώστε να αποθηκευτεί η τιμή, όταν έχουμε την εντολή `printout t` είναι αναγκαίο να έχουμε μια συμβολοσειρά προς εμφάνιση αλλίως είναι άσκοπη η χρήση της εντολής. Ακουλουθεί απόσπασμα των αντίστοιχων κανόνων παραγωγής:

```
.

.

.

| BIND ARGUMENTS PARENTHESI1 expr PARENTHESI2 { $2 = $4; $$ = $2; fprintf(yyout,"\n\tLINE %d:
ERROR! BIND XWRIS METAVLITI\n",line); lathosleks++; lathosekf++; }

       | BIND VARIABLE expr PARENTHESI2 { $$ = $2; fprintf(yyout,"\n\tLINE %d: ERROR! BIND
XWRIS DEXSIA PARENTHESI\n",line); lathosekf++; }

.

.

.

       | PARENTHESI1 DEFFACTS defexpr PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR! DEFFACTS
XWRIS ONOMA\n",line); lathosekf++; }

       | PARENTHESI1 DEFFACTS orismata  PARENTHESI2 { fprintf(yyout,"\n\tLINE %d: ERROR!
DEFFACTS XWRIS EXPRESSIONS\n",line); lathosekf++; }

.

.

.

| PARENTHESI1 PRINTOUT PARENTHESI2  { fprintf(yyout,"\n\tLINE %d: ERROR! PRINTOUT XWRIS GEGONOTA
PROS EKTYPWSH\n",line); lathosekf++; }

.

.

.
```

Στο αρχείο με τον κώδικα flex οι κανονικές εκφράσεις που προστέθηκαν για την διαχείριση λανθασμένων λεξημάτων δεν εμφάνιζαν το αναμενόμενο μήνυμα λάθους και κατά την μεταγλώττιση εμφάνιζαν προειδοποιήσεις. Για τον λόγω αυτό βρίσκονται σε σχόλια μέσα στον κώδικα. Η μεταγλώττιση παρόλα αυτά ήταν επιτυχής (αν εξαιρέσουμε τις προειδοποιήσεις) και με τις κανονίκες εκφράσεις για την διαχείριση λανθασμένων λεξημάτων.

ΕΞΑΝΤΛΗΤΙΚΟΣ ΕΛΕΓΧΟΣ ΛΑΝΘΑΣΜΕΝΩΝ ΕΝΤΟΛΩΝ

Έλεγχος εντολών υπολογισμού πράξεων ακεραίων (δύο ή περισσότερους ακέραιους) :

Εντολές:

```
+ 5 6

- 6 6 4

* 5

5 + 5

6 / 3

4 = 4

3 4

- 3 4

* 4 5

/ 5 6 7

(3 4)
```

Δίνω τις παραπάνω εντολές στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εμφάνισης κατάλληλα διαμορφωμένου μηνύματος για την κάθε εντολή ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί το ανάλογο μήνημα λάθους.

Αποτέλεσμα:

```
LINE 51: Flex -> Matched token: +

 LINE 51: Flex -> Matched token: 5

 LINE 51: Flex -> Matched token: 6


LINE 51: ERROR! PROSTHESI XWRIS PARENTHESEIS

 LINE 52: Flex -> Matched token: -

 LINE 52: Flex -> Matched token: 6

 LINE 52: Flex -> Matched token: 6

 LINE 52: Flex -> Matched token: 4


LINE 52: ERROR! AFAIRESH XWRIS PARENTHESEIS
```

Τα παραπάνω αποτελέσματα είναι για περιπτώσεις πιθανών λαθών για το συντακτικό της mini-CLIPS. Τα επόμενα δεν εμφανίζουν προειδοποιητικό μήνυμα δίοτι θεωρήσαμε πως ένας προγραμματιστής που χρησημοποιεί τη γλώσσα αυτή θα ξέρει ότι στις πράξεις οι τελεστές βρήσκονται πριν τους αριθμούς προς υπολογισμό.

```
 LINE 53: Flex -> Matched token: *

 LINE 53: Flex -> Matched token: 5

 LINE 54: Flex -> ERROR, lexeme '

' : syntax error

ERROR !

 LINE 54: Flex -> Matched token: 5
```

```
      LINE 54: Flex -> Matched token: +
      LINE 54: Flex -> Matched token: 5
      LINE 55: Flex -> ERROR, lexeme '
' : syntax error
     ERROR !
      LINE 55: Flex -> Matched token: 6
      LINE 55: Flex -> Matched token: /
      LINE 55: Flex -> Matched token: 3
      LINE 56: Flex -> ERROR, lexeme '
' : syntax error
     ERROR !
      LINE 56: Flex -> Matched token: 4
      LINE 56: Flex -> Matched token: =
      LINE 56: Flex -> ERROR, lexeme '=' : syntax error
      LINE 56: Flex -> Matched token: 4
     ERROR !
```

Τα παρακάτω αποτελέσματα είναι μερικά ακόμα από τα πιο πιθαμά λάθη της σύνταξης της γλώσσας:

```
LINE 60: Flex -> Matched token: (
      LINE 60: Flex -> Matched token: 5
```

```
 LINE 60: Flex -> Matched token: )
 LINE 61: Flex -> Matched token: 3
 LINE 61: Flex -> Matched token: 4

LINE 61: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS
 LINE 62: Flex -> Matched token: -
 LINE 62: Flex -> Matched token: 3
 LINE 62: Flex -> Matched token: 4

LINE 62: ERROR! AFAIRESH XWRIS PARENTHESEIS
 LINE 63: Flex -> Matched token: *
 LINE 63: Flex -> Matched token: 4
 LINE 63: Flex -> Matched token: 5

LINE 63: ERROR! POLLAPLASIAMOS XWRIS PARENTHESEIS
 LINE 64: Flex -> Matched token: /
 LINE 64: Flex -> Matched token: 5
 LINE 64: Flex -> Matched token: 6
 LINE 64: Flex -> Matched token: 7

LINE 64: ERROR! DIAIRESH XWRIS PARENTHESEIS
 LINE 65: Flex -> Matched token: (
 LINE 65: Flex -> Matched token: 3
 LINE 65: Flex -> Matched token: 4
```

```
      LINE 65: Flex -> Matched token: )
```

```
     LINE 64: ERROR! DEN YPARXEI SIMVOLO KAPOIAS PRAXSIS
```

Έλεγχος εντολών δήλωσης γεγονότων**:**

Εντολές:

```
(deffacts (packman-at))

(deffacts dynamic6 pack))

(deffacts dynamic6 (pack)

deffacts dynamic6 (pack))

(deffacts dynamic6 (pack)
```

Δίνω τις παραπάνω εντολές στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εμφάνισης κατάλληλα διαμορφωμένου μηνύματος για την κάθε εντολή ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί το ανάλογο μήνημα λάθους.

Αποτέλεσμα:

```
LINE 66: Flex -> Matched token: (
      LINE 66: Flex -> Matched token: deffacts
      LINE 66: Flex -> Matched token: (
      LINE 66: Flex -> Matched token: packman-at
      LINE 66: Flex -> Matched token: )
```

```
 LINE 66: Flex -> Matched token: )


LINE 66: ERROR! DEFFACTS XWRIS ONOMA
 LINE 67: Flex -> Matched token: (
 LINE 67: Flex -> Matched token: deffacts
 LINE 67: Flex -> Matched token: dynamic6
 LINE 67: Flex -> Matched token: pack
 LINE 67: Flex -> Matched token: )


LINE 67: DEFFACTS
 LINE 67: Flex -> Matched token: )
 LINE 67: Flex -> ERROR, lexeme ')' : syntax error
ERROR !
 LINE 68: Flex -> Matched token: (
 LINE 68: Flex -> Matched token: deffacts
 LINE 68: Flex -> Matched token: dynamic6
 LINE 68: Flex -> Matched token: (
 LINE 68: Flex -> Matched token: pack
 LINE 68: Flex -> Matched token: )


LINE 68: ERROR! DEFFACTS XWRIS DEXSIA PARENTHESI
 LINE 69: Flex -> Matched token: deffacts
 LINE 69: Flex -> Matched token: dynamic6
 LINE 69: Flex -> Matched token: (
```

```
 LINE 69: Flex -> Matched token: pack
 LINE 69: Flex -> Matched token: )
 LINE 69: Flex -> Matched token: )

LINE 69: ERROR! DEFFACTS XWRIS ARISTERH PARENTHESI
 LINE 70: Flex -> Matched token: (
 LINE 70: Flex -> Matched token: deffacts
 LINE 70: Flex -> Matched token: dynamic6
 LINE 70: Flex -> Matched token: (
 LINE 70: Flex -> Matched token: pack
 LINE 70: Flex -> Matched token: )

LINE 70: ERROR! DEFFACTS XWRIS DEXSIA PARENTHESI
```

Τα μηνύματα λάθους που εμφανίζονται είναι τα αναμενόμενα.

Έλεγχος εντολών δήλωσης κανόνων**:**

Εντολές:

```
defrule MOVE-UP (packman-at ?x ?y))
(defrule MOVE-UP (packman-at ?x ?y)
(defrule (packman-at ?x ?y))
(defrule MOVE-UP)
```

Δίνω τις παραπάνω εντολές στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εμφάνισης κατάλληλα διαμορφωμένου μηνύματος για την κάθε εντολή ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί το ανάλογο μήνημα λάθους.

Αποτέλεσμα:

```
LINE 71: Flex -> Matched token: defrule

      LINE 71: Flex -> Matched token: MOVE-UP

      LINE 71: Flex -> Matched token: (

      LINE 71: Flex -> Matched token: packman-at

      LINE 71: Flex -> Matched token: ?x

      LINE 71: Flex -> Matched token: ?y

      LINE 71: Flex -> Matched token: )

      LINE 71: Flex -> Matched token: )


     LINE 71: ERROR! DEFRULE XWRIS ARISTERH PARENTHESI

      LINE 72: Flex -> Matched token: (

      LINE 72: Flex -> Matched token: defrule

      LINE 72: Flex -> Matched token: MOVE-UP

      LINE 72: Flex -> Matched token: (

      LINE 72: Flex -> Matched token: packman-at

      LINE 72: Flex -> Matched token: ?x

      LINE 72: Flex -> Matched token: ?y

      LINE 72: Flex -> Matched token: )


     LINE 72: ERROR! DEFRULE XWRIS DEXSIA PARENTHESI
```

```
 LINE 73: Flex -> Matched token: (
 LINE 73: Flex -> Matched token: defrule
 LINE 73: Flex -> Matched token: (
 LINE 73: Flex -> Matched token: packman-at
 LINE 73: Flex -> Matched token: ?x
 LINE 73: Flex -> ERROR, lexeme '?x' : syntax error
 LINE 73: Flex -> Matched token: ?y
 LINE 73: Flex -> Matched token: )
 LINE 73: Flex -> Matched token: )
ERROR !
 LINE 74: Flex -> Matched token: (
 LINE 74: Flex -> Matched token: defrule
 LINE 74: Flex -> Matched token: MOVE-UP
 LINE 74: Flex -> Matched token: )

LINE 74: ERROR! DEFRULE XWRIS EXPRESSIONS
 LINE 75: Flex -> Matched token: (
 LINE 75: Flex -> Matched token: printout t
 LINE 75: Flex -> Matched token: )

LINE 75: ERROR! PRINTOUT XWRIS GEGONOTA PROS EKTYPWSH
```

Τα μηνύματα λάθους που εμφανίζονται είναι τα αναμενόμενα.

Έλεγχος εντολών printout, test και bind:

Εντολές:

```
(printout t)

printout t(ektypesh))

(printout t(ektypesh)

(test)

test ?k 3)

(test ?k 3

bind ?x 5)

bind c 5

(bind kati 5)

(printout t)
```

Δίνω τις παραπάνω εντολές στον συντακτικό αναλυτή για να ελέγξω την ορθότητα της εμφάνισης κατάλληλα διαμορφωμένου μηνύματος για την κάθε εντολή ως προς το συντακτικό και την εκτέλεση. Το αναμενόμενο αποτέλεσμα εκτέλεσης της εντολής είναι να εμφανιστεί το ανάλογο μήνημα λάθους.

Αποτέλεσμα:

```
LINE 76: Flex -> Matched token: printout t

      LINE 76: Flex -> Matched token: (

      LINE 76: Flex -> Matched token: ektypesh

      LINE 76: Flex -> Matched token: )

      LINE 76: Flex -> Matched token: )
```

```
LINE 76: ERROR! PRINTOUT XWRIS ARISTERH PARENTHESI
 LINE 77: Flex -> Matched token: (
 LINE 77: Flex -> Matched token: printout t
 LINE 77: Flex -> Matched token: (
 LINE 77: Flex -> Matched token: ektypesh
 LINE 77: Flex -> Matched token: )

LINE 77: ERROR! PRINTOUT XWRIS DEKSIA PARENTHESI
 LINE 78: Flex -> Matched token: (
 LINE 78: Flex -> Matched token: test
 LINE 78: Flex -> Matched token: )

LINE 78: FACTS
 LINE 79: Flex -> Matched token: test
 LINE 79: Flex -> ERROR, lexeme 'test' : syntax error
 LINE 79: Flex -> Matched token: ?k
 LINE 79: Flex -> Matched token: 3
 LINE 79: Flex -> Matched token: )
ERROR !
 LINE 80: Flex -> Matched token: (
 LINE 80: Flex -> Matched token: test
```

```
      LINE 80: Flex -> Matched token: ?k

      LINE 80: Flex -> Matched token: 3

      LINE 81: Flex -> ERROR, lexeme '

' : syntax error

     ERROR !

      LINE 81: Flex -> Matched token: bind

      LINE 81: Flex -> ERROR, lexeme 'bind' : syntax error

      LINE 81: Flex -> Matched token: ?x

      LINE 81: Flex -> Matched token: 5

      LINE 81: Flex -> Matched token: )

     ERROR !

      LINE 82: Flex -> Matched token: bind

      LINE 82: Flex -> ERROR, lexeme 'bind' : syntax error

      LINE 82: Flex -> ERROR, lexeme 'c' : Den anagnoristike to
lexima.

      LINE 82: Flex -> Matched token: (

      LINE 82: Flex -> Matched token: bind

      LINE 82: Flex -> Matched token: kati

      LINE 82: Flex -> Matched token: 5

      LINE 82: Flex -> Matched token: )

     ERROR !

      LINE 83: Flex -> Matched token: (
```

```
 LINE 83: Flex -> Matched token: printout t

 LINE 83: Flex -> Matched token: )


LINE 83: ERROR! PRINTOUT XWRIS GEGONOTA PROS EKTYPWSH

PLITHOS SWSTWN LEKSEWN: 98

PLITHOS SWSTWN EKFRASEWN: 18


PLITHOS LANTHASMENWN LEKSEWN: 0

PLITHOS LANTHASMENWN EKFRASEWN: 37
```

Τα μηνύματα λάθους που εμφανίζονται είναι τα αναμενόμενα.

Σημείωμα με παρατηρήσεις-προβλήματα που πρέπει να προσέξει ο αξιολογητής της εργασίας.

A3 εργασία:

- Για την εκτέλεση του προγράμματος στο τερματικό (σε αυτό το μέρος μόνο) δίνουμε την εντολή «./Makefile».

B1 εργασία:

- 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 ακεραίους.
- Οι δεσμευμένες λέξεις είναι δεκτές με πεζούς και κεφαλαίους χαρακτήρες (δεν είναι επιθυμητό από την περιγραφή της mini-CLIPS).
- Στην παρούσα εργασία η εισαγωγή των εντολών στον συντακτικό αναλυτή γίνεται με επιτυχία από το αρχείο input.txt αλλά δεν αποθηκεύονται οι ορθές εντολές στο αρχείο output.txt. Θα διορθωθεί σε επόμενη έκδοση του συντακτικού αναλυτή.
- Η ανάγνωση από το αρχείο input.txt γίνεται με ανακατεύθυνση (μόνο σε αυτό το μέρος).
- Η εκτέλεση των εντολών δεν εμφανίζει μήνυμα που πιστοποιεί την ορθή εκτέλεση της λειτουργίας (για παράδειγμα μετά την εκτέλεση της `read` δεν εμφανίζεται μήνυμα στην οθόνη ότι έγινε είσοδος ενός στοιχείου από το πληκτρολόγιο).
- Σε αυτό το μέρος τα σχόλια υλοποιήθηκαν αλλά δεν αναγνωρίζονται σωστά από το συντακτικό αναλυτή (αντί να αγνοούνται εμφανίζεται ένας μεγάλος ακέραιος αριθμός).
- Έιχαμε πάρει περιπτώσεις για κάθε μια ξεχωριστά και όλοι οι κανόνες ήταν μέσα στο `expr,` λάθος το οποίο διορθώθηκε στις εργασίες B2,B3.
- Δεν υλοποιήθηκε η μέθοδος πανικού σε αυτό το μέρος.
- Δεν υλοποιήθηκαν όλες οι εκφράσεις σε αυτό το μέρος (κάτι που γίνεται στις επόμενες εργασίες).

B2 εργασία:

- Η καταμέτρηση λανθασμένων και σωστών λεξημάτων-εντολών υλοποιήθηκε αλλά δεν είναι σωστή Δεν είναι σωστό το πλήθος σωστών εκφράσεων γιατί ο κώδικας bison δέχεται πολλά ορίσματα ως ένα ενιαίο, λόγου χάρη στις γραμμές 39-42 διαβάζει τις εντολές ως μια. Δεν γνωρίζαμε πως να υλοποιήσουμε τις λάθος λέξεις και πως να τις ορίσουμε στον κώδικα bison και flex.
- Σε αυτό το μέρος τα σχόλια υλοποιήθηκαν αλλά δεν αναγνωρίζονται από το συντακτικό αναλυτή σωστά.
- Η υλοποίηση του defrule δεν είναι όπως ζητήται από την περιγραφή της mini-CLIPS, έχει χωριστεί σε κομμάτια (αντί για μια μεμονομένη εντολή είναι από δύο και περισσότερες). Ετσί είναι και στην επόμενη έκδοση του συντακτικού αναλυτή μας.

B3 εργασία:

- Η καταμέτρηση λανθασμένων και σωστών λεξημάτων-εντολών υλοποιήθηκε αλλά δεν είναι σωστή Δεν είναι σωστό το πλήθος σωστών εκφράσεων γιατί ο κώδικας bison δέχεται πολλά ορίσματα ως ένα ενιαίο, λόγου χάρη στις γραμμές 39-42 διαβάζει τις εντολές ως μια. Δεν γνωρίζαμε πως να υλοποιήσουμε τις λάθος λέξεις και πως να τις ορίσουμε στον κώδικα bison και flex. (έγινε προσπάθεια διόρθωσης αλλά πάλι υπάρχει θέμα).
- Σε αυτό το μέρος τα σχόλια υλοποιήθηκαν αλλά δεν αναγνωρίζονται από το συντακτικό αναλυτή σωστά (εμφανίζεται ένας μεγάλος ακέραιος αριθμός).
- Πιθανότατα η υλοποίηση των κανόνων παραγωγής για τα συχνά συντακτικά λάθη να μην είναι πλήρης (λόγω χρόνου μπορεί να μην τα ανακαλύψαμε όλα).
- Στο αρχείο flex προσπαθήσαμε να υλοποιήσουμε τις κανονικές εκφράσεις για την αναγνώριση των πιο συχνά λανθασμένων λεξημάτων αλλά δεν δούλευε και δεν ξέραμε πως να το διορθώσουμε. (Είναι σε σχόλια μέσα στο αρχείο δίοτι μας έβγαζαν warnings).

Σημείωμα με αλλαγές που έγιναν σε κάποια μέρη

B2 εργασία:

- Αλλάξαμε τους κανόνες παραγωγής από τον κώδικα bison της εργασίας B1 ώστε να είναι αναδρομικοί όπου απαιτήται και να είναι χωρισμένοι για κάθε διαφορετική περίπτωση (όχι να είναι όλα μέσα στο `expr`).
- Οι εντολές για σύγκριση ή πράξεις αριθμητικές μπορούν πλεον να δεχθούν από δύο ή περισσότερους αριθμούς.
- Οι δεσμευμένες λέξεις είναι δεκτές ΜΟΝΟ με πεζούς χαρακτήρες.
- Στην παρούσα έκδοση η εισαγωγή των εντολών στον συντακτικό αναλυτή από αρχείο διορθώθηκε και λειτουργεί ορθά.
- Η εκτέλεση των εντολών εμφανίζει μήνυμα που πιστοποιεί την ορθή εκτέλεση της λειτουργίας.
- Οι κανόνες παραγωγής είναι χωρισμένοι κατάλληλα.
- Υλοποιήθηκε η μέθοδος πανικού σε αυτό το μέρος.
- Υλοποιήθηκαν σχεδόν όλες οι εκφράσεις σε αυτό το μέρος.