Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Μεταγλωττιστές

Δεύτερη Σειρά Ασκήσεων

19 Δεκεμβρίου 2018

(παράδοση: *τέλος εξεταστικής Ιανουαρίου ή Σεπτεμβρίου*)

Άσκηση 1:

Η παρακάτω γραμματική χωρίς συμφραζόμενα παράγει δηλώσεις για ένα αναγνωριστικό:

decl	→ DECLARE ID option_list
option_list	→ option_list option | ε
option	→ mode | scale | precision | storage
mode	→ INTEGER | REAL | COMPLEX
scale	→ FIXED | FLOATING
precision	→ BYTE | HALF | SINGLE | DOUBLE
storage	→ STATIC | GLOBAL | AUTO | REGISTER

όπου με κεφαλαία παριστάνονται τα τερματικά σύμβολα της γραμματικής. Για τα υπόλοιπα σύμβολα: decl είναι η δήλωση, option_list είναι μια λίστα χαρακτηριστικών και option είναι ένα χαρακτηριστικό του αναγνωριστικού. Τέσσερα χαρακτηριστικά καθορίζονται από τα μη τερματικά σύμβολα: mode, scale, precision και storage, όπου mode είναι ο τύπος του αναγνωριστικού, scale είναι ο τρόπος αναπαράστασης της τιμής του, precision είναι η ακρίβεια της αναπαράστασης και storage είναι η κατηγορία αποθήκευσης του αναγνωριστικού. Κάθε χαρακτηριστικό έχει πολλαπλές επιλογές τιμής.

Α. Παρατηρήστε ότι η γραμματική επιτρέπει αποδόσεις τιμών σε χαρακτηριστικά που έχουν ξαναχρησιμοποιηθεί ή που είναι ασύμβατες με κάποιες άλλες. Για παράδειγμα, η αρχική γραμματική επιτρέπει τη δήλωση:

```
DECLARE ZAP REAL FIXED REAL INTEGER FLOATING
```

όπου υπάρχουν τρεις τιμές για το χαρακτηριστικό mode, από τις οποίες οι δύο είναι οι ίδιες, καθώς και δύο τιμές για το χαρακτηριστικό scale, ενώ οι τιμές INTEGER του χαρακτηριστικού mode και FLOATING του χαρακτηριστικού scale είναι ασύμβατες μεταξύ τους.
Αν θέλουμε να επιβάλουμε στη σύνταξη της γραμματικής να μην επιτρέπει τέτοιες δηλώσεις, μπορούμε να τροποποιήσουμε τους πιο πάνω κανόνες, ώστε (α) κάθε επιλογή να μπορεί να υπάρχει μέχρι μία φορά, χωρίς όμως προκαθορισμένη σειρά χαρακτηριστικών, και (β) να μην υπάρχουν οι ασύμβατες αποδόσεις τιμών (i) mode = INTEGER και scale = FLOATING, (ii) mode = REAL ή COMPLEX και precision = HALF ή BYTE, (iii) mode = REAL ή COMPLEX, precision = DOUBLE και storage = REGISTER.
Δώστε μια τέτοια τροποποίηση για τη γραμματική.

Β. Αντί της τροποποίησης του προηγούμενου ερωτήματος, σχεδιάστε τη σημασιολογική ανάλυση που πρέπει να εφαρμοστεί στους υπάρχοντες κανόνες για την ικανοποίηση των περιορισμών, χρησιμοποιώντας κατάλληλα κατηγορήματα ή καθολικές μεταβλητές.

Γ. Τι μπορείτε να συμπεράνετε για την επιβολή περιορισμών του παραπάνω τύπου στη σύνταξη μιας γλώσσας;

Άσκηση 2:

Οι πιο κάτω κανόνες ορίζουν τη γραμματική κάποιας γλώσσας προγραμματισμού:

```
P → DS P  |  S
DS→ D ';'  |  S ';'
D → D ';' D  |  I ':' T
I → I ',' id  |  id
T → char  |  int  |  float  |  bool
T → array AL of T  |  record D end  |  '*' T
AL→ AL '[' inum ']'  |  '[' inum ']'
S → S ';' S  |  inum S
S → if E do inum P  |  while E do inum P  |  with E do inum P
S → E ':=' E
E → E '[' E ']'  |  E '.' id  |  '*' E
E → E and E  |  E or E  |  not E
E → E '=' E  |  E '<' E  |  E '>' E
E → E '+' E  |  E '-' E  |  E '*' E  |  E '/' E  |  E '%' E
E → lit  |  inum  |  fnum  |  bcon
E → id  |  '(' E ')'
```

Ένα πρόγραμμα της γλώσσας αυτής είναι ένα μπλοκ δηλώσεων-εντολών, δηλαδή μια ακολουθία ανάμικτων δηλώσεων και εντολών, η οποία τελειώνει υποχρεωτικά σε εντολή.
Οι λέξεις-κλειδιά "char", "int", "float" και "bool" παριστάνουν τους τέσσερεις βασικούς τύπους της γλώσσας, character, integer, floating point και boolean, αντίστοιχα. Οι λέξεις-κλειδιά "array" και "of" ορίζουν την κατασκευή ενός σύνθετου τύπου πίνακα, ενώ οι λέξεις-κλειδιά "record" και "end" ορίζουν την κατασκευή ενός σύνθετου τύπου εγγραφής. Ένας τύπος πίνακα αποτελείται από στοιχεία ίδιου τύπου και δεικτοδοτείται με τη βοήθεια του τελεστή '[ ]'. Αντίθετα, ένας τύπος εγγραφής αποτελείται από πεδία πιθανά διαφορετικών τύπων που αναφέρονται με τη βοήθεια του τελεστή '.'. Ο τελεστής '*' στα αριστερά κάποιου τύπου κατασκευάζει ένα σύνθετο τύπο δείκτη προς τον τύπο αυτό.
Οι σταθερές της γλώσσας παριστάνονται στην πιο πάνω γραμματική με τα τερματικά σύμβολα "lit", "inum", "fnum" και "bcon", για καθέναν από τους τέσσερεις βασικούς τύπους character, integer, floating point και boolean, αντίστοιχα, όπου οι σταθερές lit είναι απλοί χαρακτήρες, οι σταθερές inum είναι ακέραιες σταθερές, οι σταθερές fnum είναι πραγματικές σταθερές και οι σταθερές bcon είναι οι σταθερές λέξεις-κλειδιά "true" και "false". Το σύμβολο "id" είναι το τερματικό σύμβολο των αναγνωριστικών της γλώσσας. Οι λέξεις-κλειδιά "and", "or" και "not" παριστάνουν τελεστές. Τέλος, τα σύμβολα '=', '<', '>', '+', '-', '*', '/' και '%' είναι οι υπόλοιποι τελεστές της γλώσσας, ενώ ειδικότερα, το σύμβολο '*' στα αριστερά έκφρασης δείκτη γίνεται τελεστής αποδεικτοδότησης.
Οι εντολές της γλώσσας είναι η ανάθεση και οι εντολές do. Μπροστά από κάθε εντολή υπάρχουν προαιρετικά ετικέτες με τη μορφή ακέραιων σταθερών. Η ανάθεση συντάσσεται με τη βοήθεια του συμβόλου ':='. Μια εντολή do συντάσσεται με κάποιο πρόθεμα ελέγχου, τη λέξη-κλειδί "do", μια ετικέτα και ένα μπλοκ δηλώσεων-εντολών. Το πρόθεμα ελέγχου μιας εντολής do μπορεί να είναι (α) πρόθεμα συνθήκης – η λέξη-κλειδί "if" ακολουθούμενη από κάποια έκφραση, (β) πρόθεμα επανάληψης – η λέξη-κλειδί "while" ακολουθούμενη από κάποια έκφραση, ή (γ) πρόθεμα εγγραφής – η λέξη-κλειδί "with" ακολουθούμενη από κάποια έκφραση.

Οι δηλώσεις και οι εντολές της γλώσσας ακολουθούν τους εξής σημασιολογικούς κανόνες:

1. Οι εντολές do δημιουργούν φωλιασμένες εμβέλειες, όπου οι δηλώσεις έχουν εμβέλεια μέχρι το τέλος της εντολής, επισκιάζοντας τυχόν δηλώσεις του ίδιου αναγνωριστικού σε κάποια εξωτερική εμβέλεια.
2. Κάθε δήλωση εισάγει το τύπο ενός ή περισσότερων αναγνωριστικών σε έναν πίνακα συμβόλων. Προϋπάρχουσα δήλωση του ίδιου αναγνωριστικού στην ίδια εμβέλεια αποτελεί σημασιολογικό σφάλμα.
3. Η αναδρομή στους κανόνες κατασκευής σύνθετων τύπων επιτρέπει φωλιάσματα, όπως για παράδειγμα στη δήλωση:
   ```
   x : array[10][100] of *record a,b : array[10] of float end
   ```
   Τα φωλιάσματα μπορούν να φτάσουν σε ένα μέγιστο βάθος 8 αναδρομών.

4. Ένας πίνακας μπορεί να είναι πολυδιάστατος. Το μέγιστο πλήθος διαστάσεων ενός πίνακα είναι 8. Οι πολλαπλές διαστάσεις δεν υπολογίζονται ως διαφορετικά φωλιάσματα στην αναδρομή κατασκευής σύνθετων τύπων.
5. Οι ακέραιες σταθερές που δίνονται στην κατασκευή ενός πίνακα ορίζουν το μέγεθος του πίνακα ανά διάσταση, έτσι ώστε η αναφορά στα στοιχεία του να γίνεται ξεκινώντας από το 0 για κάθε διάσταση. Ως ελάχιστη και μέγιστη τιμή κάθε σταθεράς ορίζεται το 1 και το 2^{16} αντίστοιχα.
6. Τα στοιχεία ενός πίνακα δε μπορούν να είναι τύπου πίνακα. Πίνακες πινάκων είναι απλά πολυδιάστατοι πίνακες.
7. Κατά την κατασκευή μιας εγγραφής δημιουργείται ένας τοπικός πίνακας συμβόλων για τα ονόματα και τους τύπους των πεδίων της εγγραφής. Ο αριθμός των διαφορετικών πεδίων που μπορεί να έχει μια εγγραφή είναι μέχρι 16.
8. Τα φωλιάσματα στις εντολές do – κι επομένως και στις αντίστοιχες εμβέλειες – μπορούν να φτάσουν σε ένα μέγιστο βάθος 8 εντολών.
9. Η έκφραση στο πρόθεμα συνθήκης και στο πρόθεμα επανάληψης μιας εντολής do πρέπει να είναι τύπου boolean.
10. Η έκφραση στο πρόθεμα εγγραφής μιας εντολής do πρέπει να είναι σύνθετου τύπου εγγραφής.
11. Η εμβέλεια μιας εντολής do με πρόθεμα εγγραφής αρχικοποιείται με αντιγραφή σε αυτήν του τοπικού πίνακα συμβόλων της εγγραφής.
12. Η τελευταία εντολή του μπλοκ δηλώσεων-εντολών μιας εντολής do πρέπει να έχει ετικέτα που να ταυτίζεται με την ετικέτα που υπάρχει στην εντολή do.
13. Αναθέσεις επιτρέπονται μεταξύ εκφράσεων του ίδιου τύπου, βασικού ή σύνθετου. Κατ' εξαίρεση επιτρέπεται ανάθεση από τύπο integer σε τύπο floating point. Το αριστερό μέλος μιας ανάθεσης επιτρέπεται να είναι μόνο έκφραση αναγνωριστικού, στοιχείου πίνακα, πεδίου εγγραφής ή αποδεικτοδότησης.
14. Χρήση αναγνωριστικού που δεν έχει δηλωθεί νωρίτερα – στην ίδια ή σε εξωτερική εμβέλεια – αποτελεί σημασιολογικό σφάλμα.
15. Αναφορά σε στοιχείο πίνακα γίνεται με τον τελεστή '[ ]' και επιτρέπεται μόνο αν η πρώτη έκφραση είναι τύπου πίνακα και η έκφραση ή οι εκφράσεις μέσα στις αγκύλες είναι τύπου integer. Το αποτέλεσμα είναι του τύπου των στοιχείων του πίνακα.
16. Σε κάθε αναφορά σε στοιχείο πίνακα, το πλήθος των εκφράσεων μέσα σε αγκύλες πρέπει να είναι ίσο με το πλήθος διαστάσεων του πίνακα.
17. Αναφορά σε πεδίο εγγραφής γίνεται με τον τελεστή '.' και επιτρέπεται μόνο αν η έκφραση είναι τύπου εγγραφής, οπότε το αναγνωριστικό πρέπει να είναι ένα από τα δηλωμένα πεδία της εγγραφής. Κατ' εξαίρεση, αναφορά σε πεδίο εγγραφής χωρίς τον τελεστή '.' – δηλαδή μόνο με το αναγνωριστικό του πεδίου – γίνεται στην εμβέλεια μιας εντολής do με πρόθεμα εγγραφής, και μόνο για τα πεδία της εγγραφής αυτής. Σε κάθε περίπτωση, το αποτέλεσμα είναι του τύπου του πεδίου που αναφέρεται.
18. Οι τελεστές '[ ]' και '.' έχουν την υψηλότερη και ίση μεταξύ τους προτεραιότητα και αριστερή προσεταιριστικότητα.
19. Αποδεικτοδότηση επιτρέπεται μόνο αν η έκφραση είναι τύπου δείκτη, οπότε το αποτέλεσμα είναι του τύπου που δεικτοδοτείται. Ο τελεστής αποδεικτοδότησης '*' έχει την αμέσως επόμενη προτεραιότητα μετά τους τελεστές '[ ]' και '.', και υψηλότερη από τους υπόλοιπους τελεστές.
20. Εφαρμογή των λογικών τελεστών and, or και not επιτρέπεται μόνο σε εκφράσεις τύπου boolean, με ίση προτεραιότητα και με αριστερή προσεταιριστικότητα. Το αποτέλεσμα είναι επίσης τύπου boolean.
21. Εφαρμογή των σχεσιακών τελεστών '=', '<' και '>' επιτρέπεται μόνο μεταξύ εκφράσεων τύπου integer, floating point και character. Το αποτέλεσμά τους είναι τύπου boolean.
22. Οι σχεσιακοί τελεστές έχουν υψηλότερη προτεραιότητα από τους λογικούς τελεστές.
23. Εφαρμογή των αριθμητικών τελεστών επιτρέπεται μόνο μεταξύ εκφράσεων τύπου integer και floating point. Κατ' εξαίρεση, ο τελεστής '%' δεν εφαρμόζεται σε τύπο floating point. Το αποτέλεσμα της εφαρμογής ενός αριθμητικού τελεστή θα είναι τύπου floating point, αν

στην έκφραση συμμετέχει έστω και ένας τύπος floating point, αλλιώς θα είναι τύπου integer.
24. Οι πολλαπλασιαστικοί τελεστές '*', '/', και '%' έχουν υψηλότερη προτεραιότητα από τους προσθετικούς τελεστές '+' και '-', ενώ οι τελεστές ίσης προτεραιότητας εφαρμόζονται με αριστερή προσεταιριστικότητα. Οι αριθμητικοί έχουν υψηλότερη προτεραιότητα από τους σχεσιακούς τελεστές.
25. Οι προσθετικοί τελεστές μπορούν να έχουν ως αριστερό τελούμενο και τύπο δείκτη, με τον περιορισμό ότι το δεξί τελούμενο είναι τύπου integer, ενώ το αποτέλεσμα της πράξης είναι του ίδιου τύπου δείκτη.
26. Μια έκφραση σε παρενθέσεις διατηρεί τον τύπο της έξω από τις παρενθέσεις.

Για την υλοποίηση σημασιολογικού αναλυτή που να εκτελεί τους αντίστοιχους σημασιολογικούς ελέγχους, η γραμματική της γλώσσας επεκτείνεται σε S-κατηγορική με τα ακόλουθα κατηγορήματα κατ' ελάχιστο:
- *name* είναι η συμβολοσειρά του ονόματος ενός αναγνωριστικού και αποτελεί συντιθέμενο κατηγόρημα του συμβόλου id.
- *value* είναι η τιμή μιας σταθεράς και αποτελεί συντιθέμενο κατηγόρημα των συμβόλων lit, inum, fnum και bcon.
- *type* είναι η δομή περιγραφής τύπου μιας έκφρασης ή δήλωσης και αποτελεί συντιθέμενο κατηγόρημα των αντίστοιχων συμβόλων E και T.

Τα δύο πρώτα από τα πιο πάνω κατηγορήματα λαμβάνουν τιμή από το λεκτικό αναλυτή, ενώ το τρίτο λαμβάνει τιμή κατά τη σημασιολογική ανάλυση.

Ο σημασιολογικός αναλυτής επικοινωνεί με ένα σύστημα πινάκων συμβόλων με τη βοήθεια των ακόλουθων συναρτήσεων:
- *create()* είναι η συνάρτηση που δημιουργεί και επιστρέφει έναν κενό πίνακα συμβόλων.
- *copy()* είναι η συνάρτηση που με παραμέτρους δύο πίνακες συμβόλων, αντιγράφει στον πρώτο όλες τις καταχωρήσεις του δεύτερου, στην τρέχουσα εμβέλεια.
- *lookup()* είναι η συνάρτηση που με παραμέτρους έναν πίνακα συμβόλων και ένα όνομα αναγνωριστικού, επιστρέφει την αποθηκευμένη καταχώρηση πλησιέστερης εμβέλειας για το αντίστοιχο αναγνωριστικό, ή NULL αν το όνομα δε βρεθεί στον πίνακα.
- *addtype()* είναι η συνάρτηση που με παραμέτρους έναν πίνακα συμβόλων, ένα όνομα αναγνωριστικού και μια δομή περιγραφής τύπου, δημιουργεί και αποθηκεύει στον πίνακα μια νέα καταχώρηση για το όνομα, επιστρέφοντας την υπάρχουσα καταχώρηση για το ίδιο όνομα στην ίδια εμβέλεια, ή NULL αν δεν υπάρχει τέτοια καταχώρηση.
- *beginscope()* και *endscope()* είναι δύο συναρτήσεις που με παράμετρο έναν πίνακα συμβόλων ξεκινούν και τερματίζουν μια εμβέλεια, αντίστοιχα.

Η ανάλυση ενός προγράμματος ξεκινάει με έναν καθολικό πίνακα συμβόλων, ενώ ένας νέος πίνακας συμβόλων δημιουργείται για κάθε τύπο εγγραφής που δηλώνεται. Μια δομή περιγραφής τύπου περιέχει τουλάχιστον έναν κωδικό τύπου. Αν πρόκειται για τύπο πίνακα, η περιγραφή περιέχει επιπλέον τις διαστάσεις και το μέγεθος ανά διάσταση, καθώς και ένα δείκτη στη δομή περιγραφής τύπου των στοιχείων του πίνακα. Αν πρόκειται για τύπο εγγραφής, η καταχώρηση περιέχει επιπλέον τον αριθμό πεδίων και ένα δείκτη στον πίνακα συμβόλων της εγγραφής. Αν πρόκειται για τύπο δείκτη, η καταχώρηση περιέχει επιπλέον ένα δείκτη στον τύπο που δεικτοδοτείται. Υποθέστε ότι διαθέτετε τρεις συναρτήσεις για την κατασκευή περιγραφών:
- *create_array()* είναι η συνάρτηση που με παραμέτρους μια σταθερά πλήθους διαστάσεων, μια λίστα μεγέθους ανά διάσταση και μια δομή περιγραφής τύπου στοιχείων, δημιουργεί και επιστρέφει τη δομή περιγραφής τύπου του αντίστοιχου τύπου πίνακα.
- *create_record()* είναι η συνάρτηση που με παράμετρο μια σταθερά πλήθους πεδίων και έναν πίνακα συμβόλων, δημιουργεί και επιστρέφει τη δομή περιγραφής τύπου του αντίστοιχου τύπου εγγραφής.
- *create_pointer()* είναι η συνάρτηση που με παράμετρο μια δομή περιγραφής τύπου, δημιουργεί και επιστρέφει τη δομή περιγραφής τύπου του αντίστοιχου τύπου δείκτη.

Α. Να μετασχηματίσετε την αρχικά διφορούμενη γραμματική, ενσωματώνοντας στη σύνταξη κάποιους από τους παραπάνω σημασιολογικούς κανόνες, ώστε η νέα γραμματική να επιλύει τα διφορούμενα στοιχεία της αρχικής και ταυτόχρονα να παράγει σημασιολογικά ορθό κώδικα.

Β. Για μετάφραση οδηγούμενη από τη σύνταξη, να χρησιμοποιήσετε τη νέα γραμματική και να γράψετε τις ρουτίνες που υλοποιούν τη σημασιολογική ανάλυση της πιο πάνω γλώσσας χρησιμοποιώντας κατάλληλα τα κατηγορήματα των συμβόλων της γραμματικής της, καθώς και τις παραπάνω συναρτήσεις. Επιλέξτε μια κωδικοποίηση τύπων, ορίστε πιθανά κι άλλα κατηγορήματα, και προσθέστε τις καθολικές μεταβλητές που κρίνετε αναγκαίες. Αριθμήστε τις σημασιολογικές ρουτίνες που γράψατε.

Γ. Θεωρήστε το πιο κάτω πρόγραμμα:

```
p : float; a : array [10][5][100] of *bool; c: bool;
b : array [1000] of float;
stp : record
      x : array [5][100] of *bool;
      y : array [50][100] of record a,b : int end
      end;
x, y, z : int;
p := 0.0;
*a[0][2][34] := true;
while not b[(y-1)%1000] < b[y%1000] do 100
     i, j : int;
     if not *stp.x[i][j] = *a[x][(y+1)%5][(z-1)%100] do 101
     101  x := y+1;
     k : array[100] of *bool;
     while c or not *(stp.x[z][stp.y[x][y].b]+j) do 200
          with stp do 201
               with y[i][j] do 202
               202  a := y[(i+1)%50][j].b + z;
          201  x[z%5] := k;
     200  y := y+1;
100  if x>y and b[z%1000] < 0.0 or *k[x] = *a[y][x][z] do 300
     300  with stp do 301
          301  *x[z][j] := *k[j];
with stp do 400
400  p := p - y[y[2*z%50][(z-1)%100].b][x].a;
```

Εφαρμόστε τη σημασιολογική ανάλυση που υλοποιήσατε, για να διαπιστώσετε αν το πρόγραμμα αυτό είναι σημασιολογικά ορθό. Πιο συγκεκριμένα, να σχεδιάσετε το δέντρο σύνταξης του προγράμματος, να σημειώσετε σε αυτό τη σειρά με την οποία εκτελούνται οι σημασιολογικές ρουτίνες από ένα συντακτικό αναλυτή τύπου LR και μαζί τον αντίστοιχο αριθμό ρουτίνας, και να βρείτε τις τιμές των συντιθέμενων κατηγορημάτων που προκύπτουν, μέχρι το τέλος του προγράμματος ή μέχρι το σημείο όπου εντοπίζετε σφάλμα.

Άσκηση 3:

Θεωρήστε την παρακάτω γραμματική των εκφράσεων κάποιας γλώσσας προγραμματισμού, παρόμοιας με τη γλώσσα C:

```
expression → expression OPER expression
expression → PREFIX expression
expression → expression POSTFIX
expression → expression '?' expression ':' expression
expression → expression '[' expression ']'
expression → expression '(' expression ')'
expression → ID
expression → CONSTANT
expression → '(' expression ')'
```

Σύμβολο τελεστή	Προσεταιριστικότητα
`[] () . -> POSTFIX ++ --`	Αριστερή
`PREFIX ++ --`	-
`PREFIX & * + - ~ !`	-
`* / %`	Αριστερή
`+ -`	Αριστερή
`<< >>`	Αριστερή
`< > <= >=`	Αριστερή
`== !=`	Αριστερή
`&`	Αριστερή
`^`	Αριστερή
`|`	Αριστερή
`&&`	Αριστερή
`||`	Αριστερή
`?:`	Δεξιά
`= *= /= %= += -= <<= >>= &= ^= |=`	Δεξιά
`,`	Αριστερή

όπου οι τελεστές περιγράφονται με τις λεκτικές μονάδες OPER, PREFIX και POSTFIX, καθώς και με τα σύμβολα "[ ]", "( )" και "? :". Στον τελευταίο κανόνα οι παρενθέσεις *δεν* αποτελούν τελεστή. Η αναλυτική προσεταιριστικότητα για όλους τους τελεστές της γλώσσας δίνεται στον πίνακα. Οι τελεστές με ένα τελούμενο δίνονται στις 3 πρώτες γραμμές μετά τη λεκτική μονάδα που τους περιγράφει. Η προτεραιότητα των τελεστών καθορίζεται από τη σειρά που αυτοί βρίσκονται στον πίνακα, από τη μεγαλύτερη προς τη μικρότερη. Τελεστές στην ίδια σειρά έχουν την ίδια προτεραιότητα. Η προσεταιριστικότητα που ορίζεται για τους τελεστές POSTFIX ++ και POSTFIX -- έχει νόημα μόνο για συνδυασμό αυτών με τους τελεστές δύο τελούμενων της ίδιας προτεραιότητας.

Α. Εξετάστε την πιο πάνω γραμματική και αναγράψτε σε αριθμημένη λίστα τις συγκρούσεις ελάττωσης-ολίσθησης ή ελάττωσης-ελάττωσης που θα εμφανιστούν κατά την ανάλυση από κάποιο συντακτικό αναλυτή τύπου LR. Δεν είναι αναγκαίο να κατασκευάσετε καταστάσεις ή πίνακα συντακτικής ανάλυσης, αν είστε εξοικειωμένοι με γραμματικές της πιο πάνω μορφής, και αρκεί να δώσετε τα στοιχεία που θα εμφανίσουν την κάθε σύγκρουση.

Β. Αν κάποιος λεκτικός αναλυτής διαχωρίζει ιδανικά τα σύμβολα, και επιστρέφει τις λεκτικές μονάδες σε κάποιον συντακτικό αναλυτή τύπου LR, δώστε το δέντρο έκφρασης που αντιστοιχεί στην ανάλυση καθεμιάς από τις πιο κάτω εκφράσεις της γλώσσας:

```
1. *a[k%4*i].g(-f->p[*u.w]=*h(a[k**t+1].m)=b+*c.i++**d.r,i)=e.r[i]->w
2. d[k[i]]=*a->n(*p.u[j],b-1)-1==0?s.j.k<<=1,x:(t,x=a[j-----i]->m[j])
3. x=(++a[i+1].f,y)?y=*b[i+2]=g(z->a[j--+2].f,**m.o)||c->k[j*3]:*b[j]
4. *a[r(n.g(b+++*u->x->t[i%k-1]*(--j,p*q[s[i+j-k]->a])))]->c=x.c->y--
```

Για απλούστευση στην εμφάνιση των δέντρων, τα φύλλα μπορούν να περιέχουν το όνομα της μεταβλητής, αν αντιστοιχούν στο σύμβολο ID, ή την τιμή της σταθεράς, αν αντιστοιχούν στο σύμβολο CONSTANT. Σημειώστε πάνω σε κάθε κόμβο που προκύπτει μετά από επίλυση σύγκρουσης τον αριθμό της σύγκρουσης αυτής από τη λίστα του προηγούμενου ερωτήματος.

Άσκηση 4:

Έστω το παρακάτω απόσπασμα κώδικα C:

```
a[i] = a[i-1] + y*c[i-1][z*d[j]->x];
b[i+1] = a[--i] + x*b[i] - z*d[j++]->z;
c[i][j] = a[i+1] - x*y*c[i][z*d[j-1]->x];
d[--j]->y = a[i++] + x*b[i] - z*d[j+1]->z;
```

όπου a και b είναι μονοδιάστατοι πίνακες ακεραίων, c είναι δισδιάστατος πίνακας ακεραίων με 100 στήλες, d είναι μονοδιάστατος πίνακας δεικτών, ενώ x, y, z, i και j είναι βαθμωτές ακέραιες μεταβλητές. Τα στοιχεία του πίνακα d δείχνουν σε δομές struct με τρία πεδία x, y και z, όλα βαθμωτού ακέραιου τύπου. Οι πίνακες a, b και c είναι δηλωμένοι ως καθολικής εμβέλειας. Οι μεταβλητές x, y και z έχουν δηλωθεί τοπικά ως στατικής αποθήκευσης. Ο πίνακας d και οι μεταβλητές i και j έχουν δηλωθεί τοπικά.
Οι δηλώσεις των πιο πάνω μεταβλητών είναι οι πρώτες που εμφανίζονται στον αντίστοιχο χώρο δεδομένων, με την σειρά που αυτές αναφέρθηκαν, χωρίς να υπολογίζουμε τις αρχικές δεσμευμένες θέσεις του εγγραφήματος δραστηριοποίησης της μονάδας που μας απασχολεί, όπου τοποθετούνται η διεύθυνση επανόδου, η τιμή αποτελέσματος και οι παράμετροι κλήσης της μονάδας.
Υποθέστε ότι οι παρενέργειες μιας έκφρασης εκτελούνται ακριβώς στο σημείο της έκφρασης όπου εμφανίζονται[1].

Α. Δώστε έναν ενδιάμεσο κώδικα σε μορφή αφηρημένου συντακτικού δέντρου (ΑΣΔ) για τον παραπάνω αρχικό κώδικα. Θεωρήστε τη συνηθισμένη γραμματική και σημασιολογία της γλώσσας C, με την τροποποίηση που προαναφέρθηκε. Σημειώστε σε κάθε κόμβο αναγνωριστικού και τελεστών ‘ [ ] ’ και ‘->’ αν η αναφορά είναι δεξιάς ή αριστερής προσπέλασης.

Β. Αν η τελική γλώσσα μετάφρασης είναι η γλώσσα MIPS, μετασχηματίστε κατάλληλα τον ενδιάμεσο κώδικα σε ΑΣΔ χαμηλότερου επιπέδου. Ειδικότερα: (α) αναπτύξτε τους κόμβους τελεστή ‘ [ ] ’ σε συνδυασμούς κόμβων πρόσθεσης και πολλαπλασιασμού, προσθέτοντας κόμβους φόρτωσης για δεξιές προσπελάσεις, (β) αναπτύξτε τους κόμβους τελεστή ‘−>’ σε συνδυασμούς κόμβων φόρτωσης και πρόσθεσης, προσθέτοντας επιπλέον κόμβους φόρτωσης για δεξιές προσπελάσεις, (γ) ενοποιήστε συνδυασμούς κόμβων, είτε για πράξεις με άμεσα τελούμενα, είτε για προσπελάσεις μνήμης με σταθερή μετατόπιση από την αρχή του αντίστοιχου χώρου δεδομένων, όταν το άμεσο τελούμενο ή η μετατόπιση έχουν κατάλληλο μέγεθος, και (δ) εφαρμόστε υποβιβασμό ισχύος σε κόμβους σύνθετων πράξεων, μετασχηματίζοντάς τους σε κόμβους ή συνδυασμούς κόμβων πιο απλών πράξεων. Υποθέστε ότι όλα τα αντικείμενα των τύπων int και pointer έχουν μέγεθος 4 bytes και ότι η τοποθέτηση αυτών στο χώρο δεδομένων είναι ευθυγραμμισμένη. Υποθέστε ακόμα ότι οι δεσμευμένες θέσεις του εγγραφήματος δραστηριοποίησης που μας απασχολεί καταλαμβάνουν χώρο 16 bytes.

Γ. Βρείτε όλες τις κοινές υποεκφράσεις στο συνολικό ενδιάμεσο κώδικα του προηγούμενου ερωτήματος, αφού μελετήσετε προσεκτικά τις αλλαγές στις διάρκειες ζωής των μεταβλητών που συμμετέχουν σε αυτόν. Ειδικότερα: (α) χρησιμοποιήστε συμβολική ανάλυση των υποεκφράσεων που περιέχουν μεταβλητές που αλλάζουν τιμή, ώστε να εντοπίσετε κοινές υποεκφράσεις που δε βρίσκονται με απλή διαπέραση του ενδιάμεσου κώδικα, και (β) εφαρμόστε διάδοση αντιγράφων όπου μπορείτε, ώστε να αποφύγετε άσκοπες φορτώσεις από τη μνήμη. Μετασχηματίστε τον ενδιάμεσο κώδικα, ώστε οι κοινές υποεκφράσεις να εμφανίζονται μόνο μια φορά. Όπου είναι απαραίτητο, μετακινήστε κατάλληλα την εφαρμογή ενός τελεστή αύξησης ή μείωσης, ώστε να διαχωρίσετε τις τιμές των εκφράσεων από τις παρενέργειές τους[2].

Δ. Προχωρήστε σε δέσμευση καταχωρητών στο συνολικό τελικό ενδιάμεσο κώδικα με τη μέθοδο χρωματισμού του γράφου αλληλεπιδράσεων, ώστε να βρείτε τον ελάχιστο αριθμό απαιτούμενων καταχωρητών. Αριθμήστε τους κόμβους με τη σειρά αποτίμησής τους από αριστερά προς τα δεξιά, και κατασκευάστε το γράφο αλληλεπιδράσεων με προσοχή, επειδή μετά την εύρεση των κοινών υποεκφράσεων, η διάρκεια ζωής κάποιων κόμβων είναι αυξημένη και ξεπερνά τα όρια των επιμέρους δέντρων έκφρασης. Θεωρήστε ότι για τη δέσμευση καταχωρητών διατίθενται οι καταχωρητές γενικού σκοπού από τον $2 έως και τον $25.

---

[1] Η κλασική C δεν ορίζει επακριβώς πότε εκτελούνται οι παρενέργειες μιας έκφρασης μεταξύ της αρχής και του τέλους της, κι έτσι μια έκφραση όπως η “++i+i” δεν έχει μονοσήμαντο αποτέλεσμα, καθώς η δεύτερη εμφάνιση του i μπορεί να δίνει είτε τη νέα είτε την παλαιά τιμή του!
[2] Για παράδειγμα, η συμβολική ανάλυση μπορεί να εντοπίσει τις εκφράσεις “p[i]” και “p[i++]” ως κοινές υποεκφράσεις, όταν εμφανίζονται με αυτή τη σειρά χωρίς ενδιάμεση αλλαγή τιμής του i, όμως για να απαλείψει τη δεύτερη, θα πρέπει να προσθέσει σε κατάλληλο σημείο την έκφραση “i++”.

Ε. Δώστε έναν τελικό κώδικα MIPS για τον παραπάνω κώδικα, παράγοντας μέχρι τρεις εντολές για κάθε κόμβο του τελικού ενδιάμεσου κώδικα, με τη σειρά αρίθμησης που κάνατε, και χρησιμοποιώντας τους καταχωρητές που βρήκατε στο προηγούμενο ερώτημα. Θεωρήστε ότι ο καταχωρητής $1 είναι διαθέσιμος για προσωρινή αποθήκευση κάποιας τιμής που χρησιμοποιείται από εντολή του ίδιου κόμβου, όταν δε μπορεί να χρησιμοποιηθεί ο δεσμευμένος καταχωρητής του κόμβου. Τέλος, θεωρήστε ότι οι καταχωρητές $28 ($gp) και $29 ($sp) περιέχουν τις αρχικές διευθύνσεις του χώρου στατικών δεδομένων και του ενεργού εγγραφήματος δραστηριοποίησης, αντίστοιχα.

Άσκηση 5:

Η γλώσσα FORTRAN υποστηρίζει μια εντολή άλματος goto που ονομάζεται αποδιδόμενο goto, με σύνταξη:

```
goto m, (x1, x2, … , xn)
```

όπου x_1, x_2, … , x_n είναι ετικέτες, οι οποίες στην FORTRAN παριστάνονται με ακέραιες σταθερές, και m είναι μια βαθμωτή ακέραια μεταβλητή.
Αν κατά την εκτέλεση της εντολής αυτής η μεταβλητή m έχει τιμή ίση με την τιμή μιας από τις ετικέτες, λαμβάνοντάς την ως ακέραια σταθερά, τότε η εκτέλεση της εντολής θα μεταφέρει τη ροή του κώδικα στη θέση της ετικέτας αυτής. Για οποιαδήποτε άλλη τιμή της m δεν εκτελείται άλμα και η ροή του κώδικα συνεχίζεται με την επόμενη εντολή.

Για παράδειγμα, στον πιο κάτω κώδικα:

```
k = 100
if (a .eq. 10) k = 102
if (a .lt. 10) k = 105
goto k,(100,102,105)
```

όπου k και a ακέραιες μεταβλητές, και 100, 102 και 105 έγκυρες ετικέτες του κώδικα, η εντολή αποδιδόμενου goto θα μεταφέρει τη ροή του κώδικα στη θέση μιας από τις ετικέτες 100, 102 και 105, ανάλογα με την τιμή της μεταβλητής a.

Θεωρήστε ότι για μια εντολή αποδιδόμενου goto ο ενδιάμεσος κώδικας παρέχει δείκτες στις εντολές που αντιστοιχούν στις ετικέτες της εντολής, με τη σειρά που οι τελευταίες δίνονται ανάμεσα στις παρενθέσεις, καθώς και τη θέση της μεταβλητής m στο χώρο δεδομένων.

Α. Δώστε ένα σχήμα μετάφρασης της εντολής αποδιδόμενου goto σε τελικό κώδικα κάποιας μηχανής καταχωρητή-καταχωρητή, όταν η γλώσσα αυτής *δεν* υποστηρίζει έμμεσο άλμα, υποστηρίζει όμως άμεσο άλμα και διακλαδώσεις.

Β. Δώστε ένα καλύτερο σχήμα μετάφρασης της εντολής αποδιδόμενου goto σε τελικό κώδικα κάποιας μηχανής καταχωρητή-καταχωρητή, όταν η γλώσσα αυτής υποστηρίζει *και* έμμεσο άλμα, ώστε ο τελικός κώδικας να εκτελεί την εντολή goto με τον ίδιο αριθμό εντολών μηχανής για όλες τις ετικέτες. Υπόδειξη: Χρησιμοποιήστε το χώρο στατικών δεδομένων για να αποθηκεύσετε τις διευθύνσεις των ετικετών.

Γ. Αν η τελική γλώσσα μετάφρασης είναι το σύνολο εντολών MIPS, εφαρμόστε τα δύο σχήματα μετάφρασης που δώσατε στην εντολή αποδιδόμενου goto του πιο πάνω παραδείγματος. Υποθέστε ότι οι ετικέτες 100, 102 και 105 του κώδικα FORTRAN απεικονίζονται στις ετικέτες L100, L102 και L105 του κώδικα συμβολικής γλώσσας, αντίστοιχα. Για απλούστευση στο πρώτο σχήμα μετάφρασης υποθέστε ότι οι διευθύνσεις των ετικετών δε βρίσκονται μακριά από τη διεύθυνση της εντολής goto. Για το δεύτερο σχήμα μετάφρασης χρησιμοποιήστε τον καταχωρητή $gp για να προσπελάσετε το χώρο στατικών δεδομένων, κάνοντας κατάλληλη υπόθεση για την απαιτούμενη μετατόπιση, και θεωρήστε ότι το μέγεθος κάθε διεύθυνσης είναι 4 bytes.

Άσκηση 6:

Θεωρήστε το πιο κάτω πρόγραμμα σε κάποια γλώσσα προγραμματισμού που υποστηρίζει φωλιασμένα υποπρογράμματα με στατικό δέσιμο:

```
program main();
    a,b: integer;
    function p(name x: integer; var y,z: integer): integer;
        a: integer;
        function f(y: integer): integer;
            b,c: integer;
            a := a+1;
            if (y <= 0) then return 2;
            else return 2*x + p(y--/2,b,c) - b*c;
        end function;
        a := b++;
        y := f(x);
        z := y*f(x) - a;
        return a;
    end function;
    a := 1;
    b := 6;
    p(a+b--,a,b);
    print(a,b);
end program.
```

Οι παράμετροι στην κλήση υποπρογραμμάτων μεταδίδονται κατ' αναφορά εάν της δήλωσής τους προηγείται η λέξη-κλειδί var, κατ' όνομα εάν της δήλωσής τους προηγείται η λέξη-κλειδί name ή κατ' αξία σε άλλη περίπτωση.
Κάθε συνάρτηση του παραπάνω προγράμματος σχηματίζει εγγράφημα δραστηριοποίησης, το οποίο περιέχει με τη σειρά (α) το σύνδεσμο προσπέλασης, (β) τη διεύθυνση επανόδου, (γ) την τιμή αποτελέσματος, (δ) τις τυπικές παραμέτρους και (ε) τις τοπικές μεταβλητές. Οι τυπικές παράμετροι και οι τοπικές μεταβλητές καταλαμβάνουν θέσεις με τη σειρά που εμφανίζονται στον κώδικα. Η αποτίμηση των εκφράσεων του κώδικα γίνεται αυστηρά από αριστερά προς τα δεξιά, με τις όποιες παρενέργειες να εκτελούνται ακριβώς στο σημείο όπου εμφανίζονται. Η επίλυση των μη τοπικών αναφορών γίνεται με τη μέθοδο του συνδέσμου προσπέλασης.

Α. Να δώσετε τον ενδιάμεσο κώδικα σε μορφή ΑΣΔ για το κυρίως πρόγραμμα και τις δύο συναρτήσεις του παραπάνω αρχικού κώδικα. Αναλύστε την εντολή if κατά τα γνωστά, ενώ θεωρήστε ότι η εντολή return δημιουργεί κόμβο τύπου return_stmt με μοναδικό παιδί τον κόμβο της έκφρασης-ορίσματος της εντολής. Η print() είναι συνάρτηση βιβλιοθήκης Ε/Ε με παραμέτρους κατ' αξία.

Β. Μια υλοποίηση περάσματος παραμέτρου κατ' όνομα είναι με πέρασμα δύο στοιχείων: (α) του περιβάλλοντος στο οποίο γίνονται οι αποτιμήσεις της παραμέτρου, δηλαδή ενός δείκτη στην αρχή του εγγραφήματος δραστηριοποίησης του καλούντος περιβάλλοντος, και (β) κώδικα με τον οποίο γίνονται οι αποτιμήσεις αυτές, δηλαδή μιας διεύθυνσης κατάλληλου υποπρογράμματος που εκτελεί την αποτίμηση στο περιβάλλον αυτό. Έτσι, για την παράμετρο κατ' όνομα του παραπάνω κώδικα αποσυνδέστε τα υποδέντρα υπολογισμού των αντίστοιχων πραγματικών παραμέτρων και δημιουργήστε ανεξάρτητα ΑΣΔ υλοποίησης αυτών των υποπρογραμμάτων. Αντικαταστήστε τις πραγματικές παραμέτρους του αρχικού ενδιάμεσου κώδικα με κόμβους κλήσης του αντίστοιχου υποπρογράμματος, όπου σαν παράμετρος θα περνάει ο πιο πάνω δείκτης.

Γ. Προχωρήστε σε απλή δέσμευση καταχωρητών στον ενδιάμεσο κώδικα του προηγούμενου ερωτήματος για παραγωγή τελικού κώδικα MIPS. Θεωρήστε ότι διατίθενται όλοι οι καταχωρητές γενικού σκοπού από τον $2 έως και τον $25. Όμως, οι καταχωρητές $4 έως $7 πρέπει να χρησιμοποιούνται για πέρασμα παραμέτρων, ο καταχωρητής $2 πρέπει να χρησιμοποιείται για επιστροφή αποτελέσματος, ενώ ο καταχωρητής $3 πρέπει να χρησιμοποιείται για πέρασμα του συνδέσμου προσπέλασης κατά την κλήση μιας συνάρτησης. Οι καταχωρητές $16 έως $23 πρέ-

πει να διατηρούν την τιμή τους κατά την κλήση μιας συνάρτησης, οπότε αν δεσμεύονται, πρέπει να αποθηκεύονται σε βοηθητικές θέσεις στη στοίβα με την είσοδο στη συνάρτηση και να ξαναφορτώνονται πριν την έξοδο από τη συνάρτηση. Αν το μέγεθος λέξης είναι 4 bytes, ποιο είναι το μέγεθος εγγραφήματος δραστηριοποίησης για καθεμία από τις μονάδες του κώδικα, συμπεριλαμβανομένων των υποπρογραμμάτων αποτίμησης των παραμέτρων κατ' όνομα; Υπόδειξη: Αν ένας κόμβος έχει διάρκεια ζωής που περιλαμβάνει κάποια κλήση συνάρτησης, τότε είναι σκόπιμο γι' αυτόν τον κόμβο να δεσμεύεται καταχωρητής που διατηρεί την τιμή του κατά την κλήση.

Δ. Δώστε έναν τελικό κώδικα MIPS για τον παραπάνω κώδικα, χρησιμοποιώντας τους καταχωρητές που βρήκατε στο προηγούμενο ερώτημα. Όπου χρειάζεστε κάποιο βοηθητικό καταχωρητή, χρησιμοποιήστε τον $1. Μην ξεχάσετε τις εντολές διαχείρισης της στοίβας στην είσοδο και στην έξοδο των συναρτήσεων, για ενεργοποίηση και απενεργοποίηση του αντίστοιχου εγγραφήματος δραστηριοποίησης, τις εντολές διαχείρισης των συνδέσμων προσπέλασης, καθώς και τις εντολές φόρτωσης και αποθήκευσης των βοηθητικών θέσεων στη στοίβα που πιθανά χρησιμοποιείτε. Εκτός των παραπάνω καταχωρητών, υποθέστε ότι ο καταχωρητής $31 χρησιμοποιείται για αποθήκευση της διεύθυνσης επανόδου στην κλήση συναρτήσεων, ο καταχωρητής $28 χρησιμοποιείται για να δείχνει την αρχή του χώρου στατικών δεδομένων, και ο καταχωρητής $29 χρησιμοποιείται για να δείχνει την τρέχουσα κορυφή στοίβας – που είναι και η αρχή του ενεργού εγγραφήματος δραστηριοποίησης.

Ε. Τροποποιήστε τον κώδικα του προηγούμενου ερωτήματος, ώστε αντί περάσματος κατ' όνομα, η λέξη name να προσδιορίζει πέρασμα κατ' ανάγκη. Δώστε μόνο το τροποποιημένο μέρος του κώδικα.

ΣΤ. Να βρείτε τι τυπώνεται από καθέναν από τους τέσσερεις πιο πάνω κώδικες, σχεδιάζοντας με το χέρι τη στοίβα εκτέλεσης ανά περίπτωση, και δείχνοντας την εξέλιξή της μέχρι τον τερματισμό της εκτέλεσης του κώδικα.

Ζ (προαιρετικά). Να επαληθεύσετε τα αποτελέσματα του προηγούμενου ερωτήματος, περνώντας τους τέσσερεις κώδικες από κάποιον προσομοιωτή MIPS (για παράδειγμα τον SPIM ή τον MARS), αφού προσθέσετε τον κώδικα για την κλήση του κυρίως προγράμματος, και τροποποιήστε τον κώδικα κλήσης της συνάρτησης print() για πραγματική εκτύπωση ανάλογα με τον προσομοιωτή. *Να παραδώσετε ηλεκτρονικά* τους τελικούς κώδικες MIPS αναφέροντας και ποιον προσομοιωτή χρησιμοποιήσατε.

Άσκηση 7-8:

Θεωρήστε την παρακάτω γραμματική των εντολών μιας απλής γλώσσας προγραμματισμού ενός αριθμητικού υπολογιστή τσέπης (calculator):

```
statement   → FUNCTION ID idlist ENTER
statement   → RETURN expression ENTER
statement   → ENDFUNCTION ENTER
statement   → WHILE expression ENTER
statement   → ENDWHILE ENTER
statement   → IF expression ENTER
statement   → ELSE ENTER
statement   → ENDIF ENTER
statement   → expression ENTER
statement   → ENTER
idlist      → idlist ID  |  ε
expression  → expression OPER expression
expression  → UNOPER expression
expression  → ID ( elist )
expression  → ID ( )
expression  → ID
expression  → CONSTANT
```

```
expression  → ( expression )
elist       → elist , expression  |  expression
```

όπου OPER είναι ένας από τους τελεστές ‘=’, ‘<’, ‘<=’, ‘>’, ‘>=’, ‘==’, ‘!=’ ‘+’, ‘-’, ‘*’, ‘/’, ‘%’, ‘^’, και UNOPER είναι ένας από τους τελεστές προσήμου ‘+’, ‘-’. Αν και η γραμματική δεν το απαγορεύει, τελεστής προσήμου δεν επιτρέπεται να βρίσκεται αμέσως πριν ή μετά από κάποιον άλλο τελεστή.

Από τους πιο πάνω τελεστές, ο τελεστής ύψωσης σε δύναμη ‘^’ έχει τη μεγαλύτερη προτεραιότητα και δεξιά προσεταιριστικότητα. Οι υπόλοιποι τελεστές έχουν προτεραιότητα και προσεταιριστικότητα όπως ορίζεται στη γλώσσα C. Όλες οι πράξεις γίνονται με ακέραιους και έχουν ακέραιο αποτέλεσμα.

Το τερματικό σύμβολο ID παριστάνει αναγνωριστικά ενός πεζού αλφαβητικού χαρακτήρα που αντιστοιχούνται σε ακέραιες μεταβλητές ή ονόματα συναρτήσεων. Οι ακέραιες μεταβλητές δε χρειάζονται δήλωση για να χρησιμοποιηθούν, και αναφορά σε μεταβλητή που δεν έχει πάρει τιμή δίνει αόριστο αποτέλεσμα. Σε κάθε εφαρμογή του τελεστή ανάθεσης ‘=’, το αριστερό μέλος πρέπει να είναι αναγνωριστικό μεταβλητής.

Το τερματικό σύμβολο CONSTANT παριστάνει μη προσημασμένες ακέραιες αριθμητικές σταθερές, οι οποίες δεν αρχίζουν με ‘0’ εκτός από τη σταθερά 0 που παριστάνεται ως “0”.

Το μη τερματικό σύμβολο statement παριστάνει μια εντολή της γλώσσας. Κάθε απλή εντολή αυτής της γλώσσας τερματίζεται με το σύμβολο αλλαγής γραμμής ENTER που αντιστοιχεί στο χαρακτήρα ‘\n’. Μια κενή εντολή περιέχει μόνο το σύμβολο ENTER.

Η γλώσσα υποστηρίζει δύο σύνθετες εντολές, την εντολή βρόχου WHILE και την εντολή διακλάδωσης IF.

Η εντολή βρόχου WHILE εκτελεί επαναληπτικά μια σειρά από εντολές που περικλείονται από τη γραμμή που περιέχει τη λέξη-κλειδί WHILE και τη γραμμή που περιέχει τη λέξη-κλειδί ENDWHILE. Οι εντολές αυτές εκτελούνται όσο η τιμή της έκφρασης που ακολουθεί τη λέξη-κλειδί WHILE είναι μη μηδενική. Μ’ άλλα λόγια, στην αρχή κάθε επανάληψης αποτιμάται η έκφραση αυτή, και αν το αποτέλεσμα είναι μηδενικό, ο βρόχος τερματίζεται και ο έλεγχος οδηγεί στην εντολή που ακολουθεί τη λέξη-κλειδί ENDWHILE, διαφορετικά οδηγεί στην εντολή που ακολουθεί την έκφραση. Μετά την εκτέλεση της εντολής που προηγείται της λέξης-κλειδί ENDWHILE, γίνεται επιστροφή στην αρχή του βρόχου με νέα αποτίμηση της έκφρασης και έλεγχο τερματισμού. Φωλιάσματα στις εντολές βρόχου δεν υποστηρίζονται, κι επομένως δεν επιτρέπεται να εμφανιστεί εντολή WHILE πριν την ENDWHILE προηγούμενου βρόχου. Φυσικά δεν επιτρέπεται να εμφανιστεί ENDWHILE χωρίς να έχει προηγουμένως εμφανιστεί η αντίστοιχη WHILE.

Η εντολή διακλάδωσης IF εκτελεί υπό συνθήκη μια σειρά από εντολές που περικλείονται από τη γραμμή που περιέχει τη λέξη-κλειδί IF και τη γραμμή που περιέχει τη λέξη-κλειδί ENDIF. Ειδικότερα, αν ανάμεσα στις δύο γραμμές *δεν* περιέχεται γραμμή με τη λέξη-κλειδί ELSE, τότε αποτιμάται η έκφραση που ακολουθεί τη λέξη-κλειδί IF, και αν η τιμή της είναι μηδενική, επόμενη εντολή που θα εκτελεστεί θα είναι η εντολή μετά τη λέξη-κλειδί ENDIF, διαφορετικά επόμενη εντολή που θα εκτελεστεί θα είναι η εντολή που ακολουθεί την έκφραση. Αν υπάρχει γραμμή με τη λέξη-κλειδί ELSE, τότε αποτιμάται η έκφραση που ακολουθεί τη λέξη-κλειδί IF, και αν η τιμή της είναι μηδενική, επόμενη εντολή που θα εκτελεστεί θα είναι η εντολή μετά τη λέξη-κλειδί ELSE, διαφορετικά επόμενη εντολή που θα εκτελεστεί θα είναι η εντολή που ακολουθεί την έκφραση, και αμέσως μετά την εντολή που προηγείται της λέξης-κλειδί ELSE θα ακολουθήσει εκτέλεση της εντολής που ακολουθεί τη λέξη-κλειδί ENDIF. Όπως και με τις εντολές βρόχου, δεν υποστηρίζονται φωλιάσματα στις εντολές διακλάδωσης, κι επομένως δε μπορεί να εμφανιστεί εντολή IF πριν την ENDIF προηγούμενης διακλάδωσης. Φυσικά δεν επιτρέπεται να εμφανιστεί ELSE ή ENDIF χωρίς να έχει προηγουμένως εμφανιστεί η αντίστοιχη IF, ούτε ELSE μετά την ENDIF που ολοκληρώνει την αντίστοιχη διακλάδωση.

Συνδυασμός βρόχων με διακλαδώσεις είναι επιτρεπτός σε ένα μόνο επίπεδο. Μια εντολή διακλάδωσης μπορεί να περιέχει οσεσδήποτε μη φωλιασμένες εντολές βρόχου, και αντίστροφα, μια εντολή βρόχου μπορεί να περιέχει οσεσδήποτε μη φωλιασμένες εντολές διακλάδωσης.

Εκτός από τις παραπάνω σύνθετες εντολές, η γλώσσα υποστηρίζει ως εντολή και τη δήλωση συνάρτησης, η οποία είναι μία σειρά από εντολές που περικλείονται από τη γραμμή που περιέχει τη λέξη-κλειδί FUNCTION και τη γραμμή που περιέχει τη λέξη-κλειδί ENDFUNCTION. Μια δήλωση συνάρτησης ξεκινάει με το όνομα της συνάρτησης και μια προαιρετική λίστα από τυπικές παραμέτρους που δίνονται μετά τη λέξη-κλειδί FUNCTION. Οι εντολές που ακολουθούν μέχρι τη γραμμή τερματισμού της δήλωσης αποτελούν το σώμα της συνάρτησης. Ανάμεσα στις εντολές αυτές μπορεί να υπάρχει εντολή επιστροφής, η οποία να δίνει και την τιμή αποτελέσματος της συνάρτησης. Μια εντολή επιστροφής αποτελείται από τη λέξη-κλειδί RETURN και την έκφραση που δίνει την τιμή αποτελέσματος. Αν κατά την εκτέλεση του κώδικα της συνάρτησης συναντηθεί τέτοια εντολή, ο έλεγχος επιστρέφει άμεσα στο σημείο κλήσης της συνάρτησης. Διαφορετικά, η επιστροφή γίνεται μετά την τελευταία εντολή του σώματος της συνάρτησης, με μηδενική τιμή αποτελέσματος. Φωλιασμένες δηλώσεις συναρτήσεων δεν επιτρέπονται.
Παρόλο που η δήλωση συνάρτησης συντακτικά είναι εντολή, δεν είναι εκτελέσιμη, αλλά παράγει έναν ενδιάμεσο κώδικα που μπορεί στη συνέχεια να εκτελεστεί με κάθε κλήση της συνάρτησης. Η αντιστοίχηση του κώδικα αυτού με το όνομα της συνάρτησης γίνεται δυναμικά με τη βοήθεια ενός ειδικού πίνακα, με μια θέση για κάθε δυνατό όνομα συνάρτησης, όπου κάθε μη μηδενική θέση δείχνει στην αρχή του κώδικα μιας από 26 το πολύ συναρτήσεις.
Η κλήση μιας συνάρτησης γίνεται μέσα από εκφράσεις, με το όνομα της συνάρτησης και μέσα σε παρενθέσεις μια προαιρετική λίστα εκφράσεων, οι τιμές των οποίων αποτελούν τις πραγματικές παραμέτρους της συνάρτησης. Σε μια κλήση συνάρτησης, οι πραγματικές παράμετροι αποτιμώνται και αντιγράφονται μία προς μία στις τυπικές, οπότε ο έλεγχος πηγαίνει στην αρχή του σώματος της συνάρτησης. Με την επιστροφή, η τιμή αποτελέσματος αντικαθιστά την κλήση μέσα στην έκφραση από όπου αυτή έγινε. Φωλιασμένες κλήσεις συναρτήσεων μπορούν να υποστηριχτούν, μέχρι ένα βάθος 16. Για το σκοπό αυτό, θα πρέπει να υλοποιείται μια στοιχειώδης στοίβα, στην οποία να τοποθετούνται δείκτες προς τα σημεία επιστροφής. Με κάθε επιστροφή, ο έλεγχος θα πηγαίνει εκεί που δείχνει ο δείκτης που μπήκε τελευταίος στη στοίβα. Κλήση συνάρτησης που δεν έχει δηλωθεί οδηγεί σε σφάλμα εκτέλεσης. Αν οι πραγματικές παράμετροι είναι λιγότερες από τις τυπικές, οι υπόλοιπες τυπικές παράμετροι λαμβάνουν μηδενική τιμή.
Ένα πρόγραμμα της παραπάνω γλώσσας αποτελείται από διαδοχικές εντολές που εκτελούνται με τη σειρά, με ροή που ακολουθεί τους παραπάνω κανόνες.

Σας ζητείται να υλοποιήσετε ηλεκτρονικά τη διερμηνεία της πιο πάνω γλώσσας, ως εξής:

7Α. Γράψτε το αρχείο εισόδου του Flex που αναγνωρίζει τα τερματικά σύμβολα της γραμματικής σε τυχαία συμβολοσειρά της γλώσσας. Προσέξτε ότι ο χαρακτήρας ‘\n’ *δεν* συγκαταλέγεται στους χαρακτήρες κενού, αλλά αποτελεί λεκτική μονάδα της γλώσσας.

7Β. Γράψτε το αρχείο εισόδου του Bison για συντακτική ανάλυση ενός προγράμματος της γλώσσας, επεκτείνοντας κατάλληλα τη γραμματική, ώστε να παράγει ακολουθίες εντολών. Εισάγετε δηλώσεις προσεταιριστικότητας των τελεστών, αφού πρώτα τους διαχωρίσετε σε διαφορετικές λεκτικές μονάδες, όπου αυτό το κρίνετε απαραίτητο.

7Γ. Προσθέστε τις σημασιολογικές ρουτίνες που εκτελούν τη σημασιολογική ανάλυση και παράγουν κάποιον ενδιάμεσο κώδικα σε μορφή ΑΣΔ. Προσέξτε ότι όλες οι εντολές παράγουν ΑΣΔ, και δεν διαχωρίζονται στατικά οι δηλώσεις συναρτήσεων από τις υπόλοιπες εντολές.

8Δ. Υλοποιήστε μια ρουτίνα διαπέρασης του δέντρου που να αποτιμά τις εκφράσεις της γλώσσας και να εκτελεί τις όποιες αναθέσεις. Ο χώρος δεδομένων ενός προγράμματος, όπου αποθηκεύονται οι τιμές των μεταβλητών, είναι ένας απλός πίνακας ακεραίων, που διευθυνσιοδοτείται από 0 έως 25, με βάση τους πεζούς αλφαβητικούς χαρακτήρες. Ο πίνακας αυτός συμπληρώνεται από τον πίνακα συναρτήσεων, επίσης 26 θέσεων, καθώς και από τη στοίβα εκτέλεσης, 16 θέσεων. Η ολοκλήρωση της δήλωσης μιας συνάρτησης αποδίδει στην κατάλληλη θέση του πίνακα συναρτήσεων ένα δείκτη προς τον κόμβο του ΑΣΔ που ξεκινά η δήλωση της συνάρτησης. Ο κόμβος αυτός θα πρέπει να περιέχει τα ονόματα των τυπικών παραμέτρων, ώστε με

κάθε κλήση της συνάρτησης να μπορούν να αντιγραφούν οι πραγματικές παράμετροι στις κατάλληλες θέσεις του χώρου δεδομένων. Από την άλλη μεριά, σε κάθε κλήση συνάρτησης, εισάγεται στη στοίβα εκτέλεσης ένας δείκτης προς τον κόμβο πατέρα του κόμβου κλήσης, εφόσον σε αυτόν γίνεται η επιστροφή του αποτελέσματος της συνάρτησης. Η ολοκλήρωση της εκτέλεσης κάθε απλής μη κενής εκτελέσιμης εντολής του αρχικού κώδικα έχει σαν αποτέλεσμα την εκτύπωση της τιμής της αντίστοιχης έκφρασης.

Δοκιμάστε τον κώδικά σας για το επισυναπτόμενο αρχείο “func.txt” και *παραδώστε τον ηλεκτρονικά.*

ΠΡΟΣΟΧΗ: Όλες οι ασκήσεις να παραδοθούν χειρόγραφες, εκτός αν αναγράφεται διαφορετικά. Κάθε χειρόγραφη άσκηση να παραδοθεί σε ξεχωριστές κόλλες από τις υπόλοιπες.