ΠΡΟΑΙΡΕΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟΥΣ ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

«ΓΛΩΣΣΑ ΓΙΑ ΜΗΧΑΝΗ ΣΤΟΙΒΑΣ»

Διδάσκων: Π. Κατσαρός Ακ. Έτος: 2007-2008

Η προαιρετική εργασία «ΓΛΩΣΣΑ ΓΙΑ ΜΗΧΑΝΗ ΣΤΟΙΒΑΣ» μπορεί να δουλευτεί σε ατομική βάση ή σε ομάδες των δύο ατόμων και το όφελος που θα έχουν οι φοιτητές ανάλογα με την περίπτωση είναι:

- απαλλαγή από την εξέταση – όπου οι φοιτητές θα προσέλθουν μόνο για να παραδώσουν την εργασία – με βαθμό ΑΡΙΣΤΑ (10) για τις ολοκληρωμένες εργασίες,
- δυνατότητα να παραδοθεί μέρος μόνο της εργασίας που αν αυτό περιλαμβάνει μόνο τη Λεξική Ανάλυση θα αποδίδει μέχρι και 25% στο βαθμό, που θα γράψετε στις εξετάσεις, ενώ αν το παραδοτέο περιλαμβάνει τη Λεξική και τη Συντακτική Ανάλυση θα αποδίδει μέχρι και 50% στο βαθμό, που θα γράψετε στις εξετάσεις.

Για να βαθμολογηθεί η εργασία, υποχρεούστε να παραδώσετε, μέχρι την ημερομηνία εξέτασης του μαθήματος ή μέχρι την ημερομηνία εξέτασης στην εξεταστική Σεπτεμβρίου. Το παραδοτέο για να είναι έγκυρο πρέπει να περιλαμβάνει:

- δίσκο (CD)
  - με τον κώδικα της «Γλώσσας για Μηχανή Στοίβας» σε πηγαία και εκτελέσιμη μορφή
  - αρχεία κειμένου ASCII με παραδείγματα έτοιμα για εκτέλεση
- αναφορά, που θα περιλαμβάνει
  - στην πρώτη σελίδα, πίνακα με τα χαρακτηριστικά, που υλοποιήσατε, στη μορφή

ΑΤΟΜΙΚΗ Ή ΟΜΑΔΙΚΗ	
ΛΕΞΙΚΗ ΑΝΑΛΥΣΗ	
ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ	
ΔΗΜΙΟΥΡΓΙΑ ΚΩΔΙΚΑ ΓΙΑ ΜΙΑ ΙΔΕΑΤΗ ΜΗΧΑΝΗ ΣΤΟΙΒΑΣ	

    που θα διευκολύνει τη βαθμολόγηση της εργασίας σας.
  - συνοπτική περιγραφή της δομής της «Γλώσσας για Μηχανή Στοίβας», που θα ολοκληρώνεται με παραδείγματα και τα αποτελέσματα, που αποδίδει η εκτέλεσή τους.
- να γίνει επίδειξη του προγράμματος στο γραφείο του διδάσκοντος

Αν ο διδάσκων κρίνει ότι το πρόγραμμα, που παραδίδετε, είναι προϊόν αντιγραφής, τότε αυτό δε λαμβάνεται υπόψη στον τελικό βαθμό.

Η «Γλώσσα για Μηχανή Στοίβας» μπορεί να υλοποιηθεί, είτε σε C, είτε σε C++. Απαραίτητα βοηθήματα είναι οι σημειώσεις – διαφάνειες και τα εργαστηριακά μαθήματα Lex Analysis Lab και Parsing Lab, που διατίθενται στη διεύθυνση delab.csd.auth.gr/~katsaros/courses.htm και οι οδηγίες χρήσης των `flex` και `byacc`, που μπορεί κανείς να βρει στη διεύθυνση www.combo.org/lex_yacc_page/.

ΛΕΞΙΚΗ ΑΝΑΛΥΣΗ

Συντάξτε μία δήλωση αναγνωριστικών (tokens) για τη λεξική ανάλυση γλώσσας που θα περιλαμβάνει:

σύμβολα (symbols)	: + * - = ( ) ,
λέξεις κλειδιά (keywords)	: let in and
ονόματα (identifiers)	: αρχίζουν από χαρακτήρα που ακολουθείται από άλλους χαρακτήρες ή ψηφία με μέγιστο μήκος οκτώ (8). Ονόματα με μεγαλύτερο μήκος πρέπει να αποκόπτονται και να παράγεται ένα προειδοποιητικό μήνυμα.
αριθμούς (numbers)	: μη προσημασμένοι ακέραιοι (unsigned integers) Αν ένας αριθμός είναι πολύ μεγάλος για να ταιριάζει σε ακέραιο τότε πρέπει να παράγεται ένα προειδοποιητικό μήνυμα και συνάρτηση λεξικής ανάλυσης να επιστρέφει την τιμή 0.

Θα πρέπει επίσης να συμπεριλάβετε ως αναγνωριστικά τα end-of-file και error. Το κενό διάστημα (space) και ο χαρακτήρας νέας γραμμής παίζουν το ρόλο διαχωριστών και δεν λαμβάνονται υπόψη ως διαχωριστές.

Για να δοκιμάσετε τη λεξική ανάλυση να αναπαραστήσετε τα ονόματα ως δείκτες σε συμβολοσειρές (μην προχωρήσετε στην δημιουργία πίνακα συμβόλων σε αυτή τη φάση). Γράψτε μία συνάρτηση main() που απλά θα τυπώνει τα αναγνωριστικά που διαβάζει στο πηγαίο πρόγραμμα.

ΠΑΡΑΔΕΙΓΜΑ

Θεωρήστε το πηγαίο πρόγραμμα

```
let inx = +(3, 4) and q3 = *(2, -5) in * (inx, q3)
```

Θα πρέπει να παραχθεί η ακόλουθη σειρά αναγνωριστικών:

<LET,> <ID, inx> <EQ,> <PLUS,> <LP> <NUM, 3> <COMMA,>
<NUM, 4> <RP,> <AND,> <ID, q3> <EQ,> <TIMES,> <LP,>
<NUM,2> <COMMA,> <MINUS,> <NUM,5> <RP> <IN,> <TIMES,>
<LP,> <ID,inx> <COMMA,> <ID,q3> <RP> <EOF>

ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ

Εφόσον έχετε ολοκληρώσει τη λεξική ανάλυση μπορείτε να προχωρήσετε στη συγγραφή της συντακτικής ανάλυσης. Η γλώσσα μας αποτελείται από εκφράσεις τύπου let όπως στο παράδειγμα

```
let x = 3 and y = +(x,2) in *(x, -y)
let x = let y = 4 in +(y,y) in *(x,x)
```

Η σύνταξη της γλώσσας περιγράφεται από τη γραμματική

```
program    →   expr eof
expr       →   let deflist in expr |
               identifier |
               number |
               plus lp expr comma expr rp |
               times lp expr comma expr rp |
               minus identfier |
               minus number
deflist    →   definition |
               definition and deflist
definition →   identifier equals expr
```

Η συντακτική ανάλυση θα κατασκευάζει ένα παράγωγο δέντρο με βάση το πηγαίο πρόγραμμα που δέχεται ως είσοδο. Το δέντρο υλοποιείται ως Αφαιρετικός Τύπος Δεδομένων (Abstract Data Type) με βάση το αρχείο κώδικα `tree.h` που επισυνάπτεται. Τα δέντρα μπορεί να αναπαριστούν εκφράσεις (expressions), ορισμούς (definitions) ή λίστες ορισμών (definition lists). Το δέντρο για ένα πρόγραμμα είναι απλά το δέντρο που αντιστοιχεί στη έκφραση που αυτό περιλαμβάνει. Τα δέντρα για ονόματα (identifiers) και αριθμούς (leaves) χρησιμοποιούνται ως φύλλα που είτε διαθέτουν δείκτη στην αρχή μιας συμβολοσειράς είτε έναν ακέραιο, ανάλογα με την περίπτωση. Τα δέντρα για εκφράσεις αθροίσματος (plus) και γινομένου (times) διαθέτουν δύο απογόνους που αντιστοιχούν στις υποεκφράσεις της πράξης. Τα δέντρα για αρνητικές εκφράσεις (minus) έχουν μόνο έναν απόγονο, που αντιστοιχεί σε δέντρο έκφρασης που περιλαμβάνει είτε ένα όνομα (identifier) είτε έναν αριθμό (number). Τα δέντρα για εκφράσεις let έχουν δύο απογόνους: ο αριστερός απόγονος αντιστοιχεί σε μία λίστα ορισμών (definition list) και ο δεξιός απόγονος σε μία έκφραση (expression). Τα δέντρα για την αναπαράσταση ορισμών ("eq trees") διαθέτουν ένα δείκτη προς την αρχή μιας συμβολοσειράς και έναν απόγονο που αντιστοιχεί σε έκφραση (expression). Τα δέντρα για λίστες ορισμών (definition lists) είτε είναι δέντρα eq trees είτε είναι κόμβοι που έχουν ως αριστερό απόγονο ένα eq tree και ως δεξί απόγονο ένα δέντρο λίστας ορισμών (definition list). Στο αρχείο κώδικα `tree.h` που επισυνάπτεται δίνονται οι δηλώσεις συναρτήσεων για την κατασκευή των προαναφερόμενων περιπτώσεων δέντρου καθώς και γι τη διάσχιση δέντρων.
Σημειώνουμε ότι πρέπει να χρησιμοποιείτε τη συνάρτηση only _child όταν προσπαθείτε να προσπελάσετε τον απόγονο μέσα σε ένα δέντρο αρνητικής έκφρασης ή ένα eq tree.

Σημειώνεται ότι οι συναρτήσεις που δέχονται δείκτες προς την αρχή συμβολοσειρών ως παραμέτρους δεν αντιγράφουν τις συμβολοσειρές και γι αυτό το λόγο πρέπει να σιγουρευτείτε ότι στις συμβολοσειρές που περνάτε ως παραμέτρους έχει εκχωρηθεί μνήμη με malloc έτσι ώστε να μην αντιγράφονται οι τιμές τους. Κατά συνέπεια στη λεξική ανάλυση όταν δεν τροποποιείται μία συμβολοσειρά θα πρέπει να γίνεται χρήση της malloc.
Η συντακτική ανάλυση μπορεί να γίνει είτε με την τεχνική προβλέπουσας αναδρομικής κατάβασης (recursive descent) είτε με τη χρήση του εργαλεία byacc. Όταν εντοπίζεται συντακτικό λάθος θα πρέπει να παράγεται ένα χρήσιμο μήνυμα του τύπου (“missing right parenthesis”, “expression cannot start with and”, “unrecognized token”) και η ανάλυση θα τερματίζει (ονόματα που δεν έχουν δηλωθεί δεν θεωρούνται συντακτικά λάθη). Αν η είσοδος αποτελείται από μία σωστή έκφραση που ακολουθείται από κάτι άλλο εκτός από end-of-file τότε πρέπει να επιστρέφεται το παράγωγο δέντρο της έκφρασης και να ένα κατάλληλο μήνυμα. Το αρχείο `interpret.c` που επισυνάπτεται σας επιτρέπει να ελέγξετε αν το παράγωγο δέντρο που κατασκευάζει ο αναλυτής σας είναι σωστό.

ΔΗΜΙΟΥΡΓΙΑ ΚΩΔΙΚΑ ΓΙΑ ΜΙΑ ΙΔΕΑΤΗ ΜΗΧΑΝΗ ΣΤΟΙΒΑΣ

Η ιδεατή μηχανή στόχος για τον μεταγλωττιστή σας είναι μία μηχανή στοίβας για λέξεις 32 bit με τέσσερις (4) καταχωρητές με τα εξής ονόματα: top (ο δείκτης της στοίβας), base, frame και temp. Τα περιεχόμενα της στοίβας αρχίζουν από τη διεύθυνση 0 και εκτείνονται με κατεύθυνση προς τις υψηλότερες διευθύνσεις. Ο καταχωρητής top δείχνει πάντα τη διεύθυνση ακριβώς πάνω από την κορυφή της στοίβας. Αρχικά τόσο στον top όσο και στον base εκχωρείται η διεύθυνση της βάσης της στοίβας. Υπάρχουν 14 εντολές από τις οποίες οι δύο δέχονται μία ακέραια παράμετρο ενώ οι υπόλοιπες δεν δέχονται καθόλου παραμέτρους. Για εκφράσεις χωρίς ονόματα χρειάζονται μόνο 4 εντολές. Αυτές είναι οι:

1	arg	Εισαγωγή του arg στη στοίβα
2		Εξαγωγή δύο τιμών από τη στοίβα και εισαγωγή του αθροίσματος
3		Εξαγωγή δύο τιμών από τη στοίβα και εισαγωγή του γινομένου
4		Εξαγωγή μιας τιμής από τη στοίβα και εισαγωγή της αντίστοιχης αρνητικής τιμής

Παραδείγματα:

Ο κώδικας για την έκφραση +(3,-4) θα ήταν
1 3 | εισαγωγή στη στοίβα του 3
1 4 | η στοίβα περιέχει τα 3 και 4
4 | η στοίβα περιέχει τα 3 και -4
2 | η στοίβα περιέχει το +(3,-4)

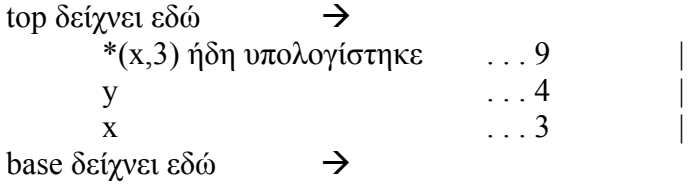
Το αποτέλεσμα είναι πάντα προσβάσιμο στη διεύθυνση που υποδεικνύει ο καταχωρητής top.

Ο κώδικας για την έκφραση *(*(2,3),+(-4,5)) θα ήταν
1 2 | εισαγωγή στη στοίβα του 2
1 3 | η στοίβα περιέχει τα 2 και 3
3 | η στοίβα περιέχει το *(2,3)
1 4 | η στοίβα περιέχει τα *(2,3), 4
4 | η στοίβα περιέχει τα *(2,3), -4
1 5 | η στοίβα περιέχει τα *(2,3), -4, 5
2 | η στοίβα περιέχει τα *(2,3), +(-4, 5)
3 | η στοίβα περιέχει το *(*(2,3), +(-4, 5))

Οι τιμές των ονομάτων περνάνε μέσα στη στοίβα με ένα stackframe για κάθε εντολή let. Ο καταχωρητής base θα δείχνει πάντα το τρέχον stack frame: τα περιεχόμενα της διεύθυνσης top θα δείχνει το επόμενο stack frame και οι μεταβλητές στο τρέχον frame θα βρίσκονται στις θέσεις που ακολουθούν μετά τη διεύθυνση που υποδεικνύει ο base.
Όταν μπαίνουμε σε ένα νέο let είναι απαραίτητο να ανοίξουμε καινούριο frame στην κορυφή της στοίβας και να ενημερώσουμε τον καταχωρητή base. Όταν εγκαταλείπουμε ένα let οι καταχωρητές base και top πρέπει να επανέλθουν στις προγενέστερες τιμές.
Έτσι, όταν εκτελούνται οι εντολές

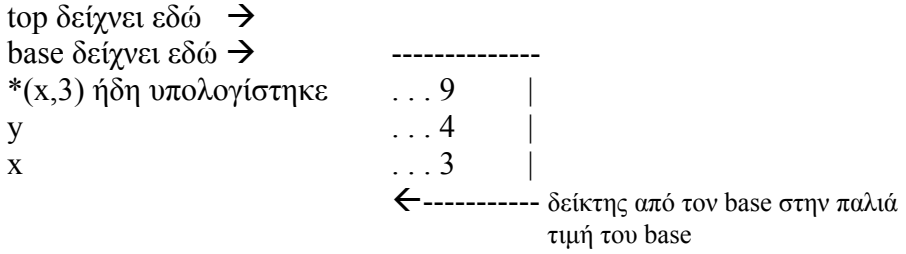
```
let x=3 and y=4 in +(*(x,3), let z=+(y,2) in *(z,3))
```

και μόλις πριν η εκτέλεση εισέλθει στο εσωτερικό let μετά την αποτίμηση των x και y η στοίβα θα έχει την παρακάτω εικόνα:



ΠΥΘΜΕΝΑΣ ΣΤΟΙΒΑΣ

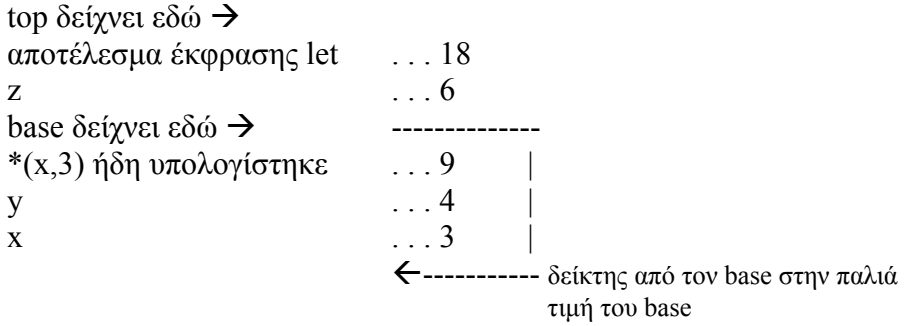
Για τη δημιουργία νέου stack frame είναι απαραίτητο να αντιγραφεί το περιεχόμενο του base στη διεύθυνση που υποδεικνύεται από τον top να ενημερωθεί η τιμή του base με τη νέα θέση και να αυξηθεί η τιμή του top κατά μία θέση. Η στοίβα θα πάρει την εξής μορφή:



ΠΥΘΜΕΝΑΣ ΣΤΟΙΒΑΣ

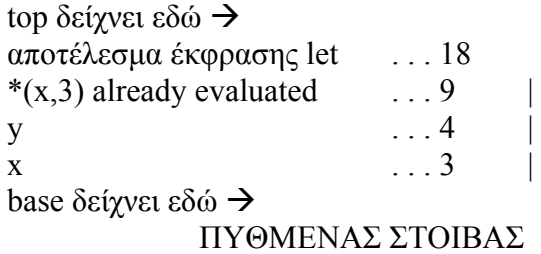
Όταν υπολογίζεται η έκφραση του εσωτερικού let τότε ο base παραμένει σταθερός ενώ μετακινείται ο top. Για να προσπελαστεί η μεταβλητή y που ανήκει στο εξωτερικό let πρέπει να γίνει προσπέλαση στον base που περιέχει δείκτη προς την προγενέστερη τιμή του base και εφόσον είναι γνωστό ότι το y είναι το δεύτερο όνομα μέσα στο let το ζητούμενο θα βρεθεί με προσπέλαση σε διεύθυνση δύο θέσεις πάνω από την παλιά διεύθυνση της base. Για να διαχειριστούμε τους δείκτες σε αυτή την περίπτωση χρησιμοποιείται ο καταχωρητής frame.

Όταν θα έχει υπολογιστεί η έκφραση του εσωτερικού let στο παράδειγμα η κατάσταση της στοίβας θα είναι όπως στο σχήμα που ακολουθεί:



ΠΥΘΜΕΝΑΣ ΣΤΟΙΒΑΣ

Για να προχωρήσουμε στη συνέχεια με την έκφραση του εξωτερικού let είναι απαραίτητο να έρθει η στοίβα στην εξής κατάσταση:



Για να γίνει αυτό, το αποτέλεσμα της έκφρασης let εξάγεται από την top και τοποθετείται στον temp, η διεύθυνση του base αντιγράφεται στον top και ο δείκτης προς την παλιά διεύθυνση του base (δηλ. το περιεχόμενο της διεύθυνσης που τώρα δείχνει ο top) αντιγράφεται στον base. Τέλος, το περιεχόμενο που προσωρινά αποθηκεύθηκε στον temp εισάγεται στην στοίβα και ενημερώνεται ο top.

Για τη δημιουργία και την καταστροφή stackframes χρησιμοποιούμε τις ακόλουθες εντολές:

5 Αυξάνεται το περιεχόμενο του top κατά 1
6 Αντιγράφεται το περιεχόμενο του top στον base
7 Αντιγράφεται το περιεχόμενο του top στον top
8 Αντιγράφεται το περιεχόμενο του top στη διεύθυνση που δείχνει ο top
9 Αντιγράφεται το περιεχόμενο της διεύθυνσης που υπάρχει στον top στον base
10 Εξάγεται ένα στοιχείο της στοίβας στον temp
11 Εισάγεται το περιεχόμενο της temp στη στοίβα

Οι υπόλοιπες εντολές απαιτούνται για την προσπέλαση στα περιεχόμενα των μεταβλητών μέσα στη στοίβα. Πρέπει να είμαστε σε θέση να αντιγράφουμε τον base στον frame, να ακολουθούμε αλυσιδωτές αναφορές προγενέστερων τιμών του base και να εισάγουμε στη στοίβα ένα αντίγραφο του περιεχομένου της διεύθυνση που προσδιορίζεται σε καθορισμένη απόσταση πάνω από μία καταχώρηση κάποιου base:

12 Αντιγράφεται το περιεχόμενο του base στον frame
13 Αντιγράφεται το περιεχόμενο της διεύθυνσης που αναφέρεται στον frame στον καταχωρητή frame
14 arg Επιστρέφεται η διεύθυνση που προκύπτει ως άθροισμα του arg και του περιεχομένου του frame. Εισάγεται το περιεχόμενο αυτής της διεύθυνσης στη στοίβα.

Το πρόγραμμα δημιουργίας κώδικα θα διασχίζει το παράγωγο δέντρο του πηγαίου προγράμματος με αναδρομικές κλήσεις για τους απογόνους κάθε κόμβου και την εκτέλεση μιας εντολής switch case που κατά περίπτωση θα λειτουργεί ως εξής:

Κόμβος Number – δημιουργείται εντολή που θα εισάγει τον αριθμό στη στοίβα

Κόμβος Identifier – δημιουργείται μία σειρά εντολών που θα εισάγει ένα αντίγραφο της τιμής του ονόματος (που βρίσκεται κάπου μέσα στη στοίβα) στη στοίβα

Κόμβος Plus – δημιουργείται κώδικας για τον αριστερό απόγονο, στη συνέχεια δημιουργείται κώδικας για τον δεξί απόγονο, στη συνέχεια δημιουργείται εντολή άθροισης (δε χρειάζεται να γίνει έλεγχος για overflow)

Κόμβος Times /Κόμβος minus κλπ.

Κόμβος Let – δημιουργείται η σειρά εντολών προλόγου, δημιουργείται κώδικας για τον αριστερό απόγονο, δημιουργείται κώδικας για τον δεξί απόγονο, δημιουργείται η σειρά εντολών επίλογου
(Η σειρά εντολών προλόγου αποτελείται από τις εντολές που απαιτούνται για το άνοιγμα ενός stackframe. Η σειρά εντολών επιλόγου αποτελείται από τις εντολές που απαιτούνται για την καταστροφή ενός stackframe και την μετακίνηση του αποτελέσματος σύμφωνα με τις ενέργειες που περιγράφηκαν στις προηγούμενες παραγράφους).

Κόμβος And – δημιουργείται κώδικας για τον αριστερό απόγονο, στη συνέχεια δημιουργείται κώδικας για τον δεξί απόγονο

Κόμβος eq – δημιουργείται κώδικας για τον μοναδικό απόγονο του κόμβου

Το πιο δύσκολο από τα προβλήματα που θα αντιμετωπίσετε είναι η σειρά των εντολών που απαιτούνται για την εύρεση της τιμής ενός συγκεκριμένου ονόματος. Για να γίνει αυτό θα πρέπει να βρεθεί η θέση του ονόματος σε σχέση με τη θέση base του let στο οποίο ορίζεται το όνομα και να υπολογιστεί η απόσταση μέσα στη στοίβα της τιμής του ονόματος από τη συγκεκριμένη θέση base.
Η διεύθυνση του base, που δεν θα ταυτίζεται με την τρέχουσα διεύθυνση στην οποία δείχνει ο καταχωρητής base αν το όνομα ορίζεται σε εντολή let διαφορετική από αυτήν που αναλύεται, θα πρέπει να αποθηκευτεί στον καταχωρητή frame. Η απόσταση της τιμής του ονόματος από τον καταχωρητή frame εξαρτάται από το πόσοι ορισμοί ονομάτων έχουν προηγηθεί μέσα στο let, δηλ. το πρώτο όνομα θα βρίσκεται μία λέξη πάνω από τη διεύθυνση του frame, το δεύτερο όνομα δύο λέξεις πάνω κ.ο.κ. Έτσι, αν θεωρήσουμε το παράδειγμα

```
let q=+(7,4) and y=let p=3 and x=+(p,4) in *(x,q) in +(q,y)
```

όταν δημιουργείται κώδικας για τον υπολογισμό του `*(x,q)` το x είναι το δεύτερο όνομα που ορίζεται στον τρέχον let και άρα για να εισαχθεί η τιμή του στη στοίβα πρέπει να εκτελεστούν οι παρακάτω εντολές:

```
12
14 2
```

Παρόλα αυτά η τιμή του q όταν υπολογίζεται το `*(x,q)` σίγουρα δεν βρίσκεται στο τρέχον stackframe αφού αυτό είχε υπολογιστεί πριν από την είσοδο στο ενεργό let. Η θέση του βρίσκεται μία λέξη πάνω από τη θέση base του προηγούμενου stackframe. Εκμεταλλευόμαστε την ύπαρξη δείκτη από την τρέχουσα θέση στη οποία δείχνει ο καταχωρητής base στην προηγούμενη θέση base και έτσι μετά από την αντιγραφή της τρέχουσας θέσης base στον καταχωρητή frame ακολουθούμε αυτόν τον δείκτη. Αυτό γίνεται με την εντολή 13 και έτσι η σειρά εντολών παίρνει την παρακάτω μορφή:

```
12
13
```

14 1

Άρα, για τη δημιουργία κώδικα για ένα όνομα χρειάζονται οι εξής πληροφορίες: (α) ο αριθμός των εμφωλευμένων εντολών let μεταξύ του ορισμού του ονόματος και της τρέχουσας θέσης, που τελικά προσδιορίζει το πόσες φορές χρειάζεται η εντολή 13 και (β) η θέση του ονόματος μέσα στη let που ορίζεται που προσδιορίζει την παράμετρο της εντολής 14.
Στη συνέχεια προτείνονται δύο προσεγγίσεις που μπορείτε να ακολουθήσετε για να παράγετε την προαναφερόμενη πληροφορία, αλλά αν εσείς θεωρείτε κάποια άλλη υλοποίηση πιο προσιτή μπορείτε να την εφαρμόσετε.
Η πρώτη προσέγγιση αποσκοπεί στην καταμέτρηση των εμφωλευμένων let που ανιχνεύονται κατά τη διάσχιση του παράγωγου δέντρου για τη δημιουργία κώδικα (δηλ. αυξάνεται κατά ένα ένας μετρητής που χρησιμοποιείται γι αυτό το σκοπό πριν από τη δημιουργία κώδικα για τον αριστερό απόγονο ενός κόμβου let και μειώνεται κατά ένα κάθε φορά που ολοκληρώνεται η δημιουργία κώδικα για ένα δεξί απόγονο κόμβου let) και στην καταμέτρηση των ονομάτων που ορίζονται μέσα στο τρέχον let. Μετά από τη δημιουργία κώδικα για τον απόγονο ενός κόμβου eq το όνομα μαζί με τους δύο προαναφερόμενους μετρητές καταχωρείται σε μία λίστα. Έτσι, για το παραπάνω παράδειγμα ενώ θα δημιουργείται κώδικας για το *(x,q) το περιεχόμενο της λίστας θα είναι το

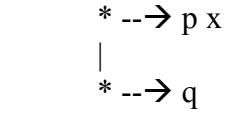
<x,2,2> <p,2,1> <q,1,1>

Για να δημιουργήσουμε κώδικα που θα αντιγράφει την τιμή του q στην κορυφή της στοίβας πρέπει πρώτα να βρούμε το q μέσα στη λίστα, να αφαιρέσουμε το επίπεδο let στο οποίο αυτό ορίζεται (και είναι το 1) από το τρέχον επίπεδο let (που είναι το 2) για να προσδιορίσουμε το πόσες εντολές 13 απαιτούνται και να χρησιμοποιήσουμε την τιμή του μετρητή των ονομάτων που είναι καταχωρημένη ως παράμετρο της εντολής 14. Μετά τη μείωση του μετρητή των let είναι απαραίτητο να απομακρύνονται τα ονόματα με παλιά μέτρηση let από τη λίστα αφού πλέον αυτά είναι εκτός εμβέλειας. Αυτό σημαίνει ότι όταν θα υπολογίζεται το +(q,y) το περιεχόμενο της λίστας θα είναι

<y,1,2> <q,1,1>

Σημειώστε ότι όταν εισέρχεστε σε ένα εμφωλευμένο let ο μετρητής ονομάτων για το εξωτερικό let κρατάει την τιμή του.

Σύμφωνα με τη δεύτερη προσέγγιση συντηρείται μία λίστα από λίστες με ονόματα. Η πρώτη στη σειρά λίστα περιέχει ονόματα που ορίζονται στην τρέχουσα εμβέλεια, η επόμενη λίστα τα ονόματα ενός επιπέδου εμβέλειας προς τα έξω κ.ο.κ. Ενώ λοιπόν δημιουργείται κώδικας για το *(x,q) η λίστα από λίστες έχει την παρακάτω μορφή



Όταν χρειάζεται να παραχθεί κώδικας προσπέλασης σε ένα όνομα πρέπει να γίνει αναζήτηση του ονόματος πρώτα στην πρώτη λίστα, μετά στην επόμενη κ.ο.κ. Έχοντας ένα μετρητή που θα καταμετρά τις λίστες στις οποίες χρειάστηκε να γίνει αναζήτηση καθορίζεται ο αριθμός των εντολών 13 που απαιτούνται και καταμετρώντας τον αριθμό των στοιχείων μέσα σε κάθε λίστα καθορίζεται και η παράμετρος της εντολής 14. Έτσι για παράδειγμα το x είναι το δεύτερο στοιχείο της πρώτης λίστας και άρα αντιστοιχεί στο δεύτερο όνομα της τρέχουσας εντολής let, ενώ το y είναι το πρώτο στοιχείο της δεύτερης λίστας και άρα το πρώτο όνομα που

εμφανίζεται στην προηγούμενη εντολή let. Είναι σημαντικό να καταστρέφεται πάντα η πρώτη από τη λίστα με τις λίστες μετά τη δημιουργία κώδικα για τον δεξί απόγονο ενός κόμβου let.

Είτε με την πρώτη είτε με τη δεύτερη προσέγγιση αν ένα όνομα δεν εντοπιστεί να παράγεται κατάλληλο μήνυμα «undeclared identifier» στην τυπική έξοδο (όχι στον κώδικα στόχο). Η διαχείριση της κατάστασης στην περίπτωση αυτή (τερματισμός προγράμματος ή απομάκρυνση κώδικα που ήδη δημιουργήθηκε ή ανάθεση προκαθορισμένης τιμής 0 στο όνομα) αφήνεται στην κρίση σας αρκεί στην αναφορά να υπάρχει σχόλιο για την επιλογή που κάνατε.

Γράψτε μία συνάρτηση δημιουργίας κώδικα που δέχεται δύο παραμέτρους, ένα αρχείο και ένα δέντρο και παράγει κώδικα για το δέντρο μέσα στο αρχείο. Οι εντολές πρέπει να εκτυπώνονται ως αριθμοί και πιο συγκεκριμένα μία εντολή ανά γραμμή.

Τώρα η συνάρτηση main θα καλεί τη συνάρτηση ανάλυσης για να κατασκευάσει το παράγωγο δέντρο από την έκφραση που θα διοχετευθεί στην τυπική είσοδο και μετά θα ανοίξει ένα αρχείο για εγγραφή με όνομα που θα περαστεί ως παράμετρος κατά την εκτέλεση του προγράμματος. Αν δεν δοθεί όνομα αρχείου λη το άνοιγμα του αρχείου που ζητήθηκε δεν ήταν επιτυχές, τότε θα πρέπει να παράγεται ένα μήνυμα λάθους και το πρόγραμμα να τερματίζει. Σε κάθε άλλη περίπτωση θα καλείται η συνάρτηση δημιουργία κώδικα περνώντας ως παραμέτρους το αρχείο και το δέντρο. Η συνάρτηση main πρέπει να κλείνει το αρχείο πριν από τον τερματισμό.

Στα αρχεία που επισυνάπτονται περιλαμβάνεται πρόγραμμα που δέχεται ως είσοδο αρχείο όπως αυτά που θα παράγονται κατά τη δημιουργία κώδικα και προσομοιώνει το αποτέλεσμα του παραγόμενου κώδικα.