Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/pl2/

Γλώσσες Προγραμματισμού II

Αν δεν αναφέρεται διαφορετικά, οι ασκήσεις πρέπει να παραδίδονται στους διδάσκοντες σε ηλεκτρονική μορφή μέσω του συνεργατικού συστήματος ηλεκτρονικής μάθησης `moodle.softlab.ntua.gr`. Η προθεσμία παράδοσης θα τηρείται αυστηρά. Έχετε δικαίωμα να καθυστερήσετε το πολύ μία άσκηση.

Άσκηση 8 Δηλωτική σημασιολογία

Προθεσμία παράδοσης: 4/3/2018

Έστω η προστακτική γλώσσα προγραμματισμού με την εντολή `while`, που ορίστηκε στην παρουσίαση της 13/12/2017 (ας την ονομάσουμε WHILE). Η δηλωτική σημασιολογία της WHILE ορίζεται στις διαφάνειες, αρχικά χρησιμοποιώντας μερικές συναρτήσεις μεταξύ συνόλων και στη συνέχεια βασισμένη στη θεωρία πεδίων (διαφάνειες 29–31). Για λόγους απλότητας, θα θεωρήσουμε ότι υπάρχουν μόνο ακέραιες μεταβλητές. Ένας διερμηνέας για τη WHILE γραμμένος σε Haskell και βασισμένος σε μία άμεση υλοποίηση της δηλωτικής σημασιολογίας είναι διαθέσιμος από τη σελίδα του μαθήματος.

Στην άσκηση αυτή, τροποποιούμε τη γλώσσα WHILE και ορίζουμε τη WHILE++, στην οποία οι εκφράσεις μπορούν να έχουν παρενέργειες. Συγκεκριμένα, η σύνταξη της WHILE++ είναι η εξής:

$$C ::= \texttt{skip} \mid N \mid C_0 \texttt{ ; } C_1 \mid \texttt{if } B \texttt{ then } C_0 \texttt{ else } C_1 \mid \texttt{for } N \texttt{ do } C \mid \texttt{while } B \texttt{ do } C$$
$$N ::= \texttt{0} \mid \texttt{succ } N \mid \texttt{pred } N \mid x \mid x \texttt{ := } N \mid x \texttt{ ++} \mid x \texttt{ --}$$
$$B ::= \texttt{true} \mid \texttt{false} \mid N_0 \texttt{ < } N_1 \mid N_0 \texttt{ = } N_1 \mid \texttt{not } B$$

Ορίστε τη δηλωτική σημασιολογία της WHILE++. Αναφέρετε τυχόν παραδοχές ή σχεδιαστικές επιλογές που κάνατε.

Στη συνέχεια, κατασκευάστε έναν διερμηνέα για τη WHILE++ στη γλώσσα προγραμματισμού της αρεσκείας σας, βασισμένο και πάλι σε μία άμεση υλοποίηση της δηλωτικής σημασιολογίας. Για δική σας διευκόλυνση, σας προτείνουμε να επιλέξετε μία εκ των Standard ML, OCaml, Haskell, Scheme, Lisp, Erlang, Prolog. Ο διερμηνέας σας θα πρέπει να εκτελεί ένα πρόγραμμα WHILE++ και να εκτυπώνει την τελική τιμή της μεταβλητής `result`.

Σημείωση. Όσοι επιλέξετε να υλοποιήσετε το διερμηνέα σε μία συναρτησιακή γλώσσα, μπορείτε να ορίσετε με εύκολο τρόπο τον τελεστή σταθερού σημείου `fix` και να τον χρησιμοποιήσετε για τον ορισμό της σημασιολογίας της εντολής `while`. Για παράδειγμα, στη Haskell (που είναι οκνηρή γλώσσα) μπορείτε να ορίσετε

```
fix f = f (fix f)          -- fix :: (t → t) → t
```

ενώ στη Standard ML (που είναι πρόθυμη)

```
fun fix f x = f (fix f) x   (* fix : (('a → 'b) → 'a → 'b) → 'a → 'b *)
```

Τι να παραδώσετε. Ένα αρχείο πηγαίου κώδικα στη γλώσσα της αρεσκείας σας. Στην αρχή πρέπει να υπάρχει ένα σχόλιο με τα στοιχεία σας και τη συγκεκριμένη υλοποίηση της γλώσσας που χρησιμοποιήσατε (όνομα μεταγλωττιστή ή διερμηνέα και έκδοση). Στο ίδιο σχόλιο πρέπει να εξηγείται ο τρόπος χρήσης του διερμηνέα σας.

Αν η σημασιολογία που ορίσατε για τη WHILE++ δεν φαίνεται άμεσα, διαβάζοντας τον κώδικα του διερμηνέα σας, φροντίστε να την περιγράψετε σε κατάλληλα σχόλια ή σε ένα ξεχωριστό έγγραφο.

Τέλος, αν η υποβολή σας περιέχει περισσότερα αρχεία, συμπιέστε τα σε ένα `zip` ή `tgz`.