Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
http://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 6 Διαχείριση μνήμης

Προθεσμία παράδοσης: 11/1/2012

Απαντήστε στις παρακάτω ερωτήσεις:

1. Ένας συμφοιτητής σας έχει σχεδιάσει μια νέα τεχνική συλλογής σκουπιδιών την οποία ονομάζει "RefCount++" και η οποία επεκτείνει την τεχνική ανακύκλωσης μνήμης μέσω μετρήματος αναφορών με την επιπλέον δυνατότητα ανίχνευσης και συλλογής μη προσβάσιμων (unreachable) κυκλικών δομών δεδομένων. Είναι πολύ περήφανος για τη συγκεκριμένη τεχνική και ισχυρίζεται ότι στα προγράμματα στα οποία η διαχείριση μνήμης γίνεται με RefCount++ δεν υπάρχουν memory leaks. Συμφωνείτε μαζί του ή όχι; Αιτιολογήστε την απάντησή σας.

2. Ένας άλλος συμφοιτητής σας θέλει να μειώσει το κόστος της αυτόματης διαχείρισης μνήμης κατά το χρόνο εκτέλεσης με το συνδυασμό στατικών και δυναμικών τεχνικών ανακύκλωσης μνήμης. Πιο συγκεκριμένα, θέλει ο compiler του να εισάγει αυτόματα όσες περισσότερες κλήσεις στη συνάρτηση `free` μπορεί κατά τη μεταγλώττιση του προγράμματος και να διαχειρίζεται τα υπόλοιπα δεδομένα μέσω κλήσεων κατά το χρόνο εκτέλεσης της συνάρτησης που υλοποιεί το συλλέκτη σκουπιδιών. Θεωρεί ότι ο παραπάνω συνδυασμός μπορεί να βασιστεί πλήρως πάνω στη ανάλυση της "ζωντάνιας" των μεταβλητών (liveness analysis) που έτσι και αλλιώς κάνει ο μεταγλωττιστής. Πιο συγκεκριμένα, προτείνει ο μεταγλωττιστής να εισάγει μια κλήση της μορφής `free(v)` στο σημείο του προγράμματος στο οποίο μια μεταβλητή `v` παύει να είναι ζωντανή. Είναι σωστή η στρατηγική του συμφοιτητή σας; Εξηγήστε το σκεπτικό της απάντησής σας.

3. Ένας άλλος συμφοιτητής σας ο οποίος είναι μεγάλος οπαδός της C++ ισχυρίζεται ότι μπορεί κάποιος να αποφύγει τα memory leaks στη C++ με το να δεσμεύει όλα τα αντικείμενα στη στοίβα. Επιπλέον σας λέει ότι αυτό είναι κάτι το οποίο δε μπορεί κάποιος να κάνει στη Java. Για το σκοπό αυτό λοιπόν προτείνει τα προγράμματα Java να μετασχηματίζονται αυτόματα σε προγράμματα C++ τα οποία χρησιμοποιούν δέσμευση μνήμης στη στοίβα. Πιο συγκεκριμένα, ο μετασχηματισμός του προτείνει ότι ένα Java statement της μορφής `Foo f = new Foo()` θα μετατρέπεται στον C++ κώδικα `Foo _f; Foo* f = &_f` ο οποίος κάνει το `f` να δείχνει σε ένα `Foo` αντικείμενο το οποίο είναι δεσμευμένο στη στοίβα αντί στο σωρό.[1] Είναι σωστός ο μετασχηματισμός αυτός; Εξηγήστε το σκεπτικό της απάντησής σας.

4. Αντίθετα με το παραπάνω, κάποιες υλοποιήσεις γλωσσών δε χρησιμοποιούν καθόλου στοίβα αλλά δεσμεύουν τα πάντα στο σωρό. Συνήθως, σε κάποια τέτοια υλοποίηση, η "στοίβα" δεν είναι τίποτε άλλο από μια απλά συνδεδεμένη λίστα από εγγραφές δραστηριοποίησης. Όταν καλείται μια συνάρτηση μια καινούρια εγγραφή δραστηριοποίησης r προστίθεται στο σωρό, η διεύθυνση της προηγούμενης εγγραφής δραστηριοποίησης ανατίθεται σε ένα πεδίο $prev$ της r (όλες οι εγγραφές

[1] Πιθανώς να πρέπει να μάθετε κάποια πράγματα για τη C++ για να το καταλάβετε αυτό.

δραστηριοποίησης έχουν από ένα τέτοιο πεδίο), και ο stack pointer δείχνει στην r. Όταν η συνάρτηση επιστρέψει ο stack pointer παίρνει την τιμή του πεδίου $prev$ και κατά συνέπεια με αυτόν τον τρόπο η προηγούμενη τρέχουσα εγγραφή δραστηριοποίησης "βγαίνει" ("pop") από τη στοίβα.

Εξετάστε τη συνολική δουλειά που πρέπει να γίνει σε κάποια τέτοια υλοποίηση για να δεσμευθεί, μπει (push), βγει (pop), και αποδεσμευθεί μια εγγραφή δραστηριοποίησης εάν ο συλλέκτης σκουπιδιών δουλεύει με τον αλγόριθμο του μαρκαρίσματος και σκουπίσματος (mark-and-sweep), και συγκρίνετε το κόστος της με το συνολικό κόστος των ίδιων ενεργειών (δηλ. δέσμευση, push, pop, και αποδέσμευση) αν οι εγγραφές δραστηριοποίησης είναι οργανωμένες σε "κλασσική" συνεχόμενη στοίβα. Είναι κάποια από τις δύο υλοποιήσεις γρηγορότερη από την άλλη ή έχουν πάνω-κάτω την ίδια επίδοση;

5. Απαντήστε την παραπάνω ερώτηση αν χρησιμοποιείται ένας stop-and-copy συλλέκτης σκουπιδιών.