HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2015-16
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 26 Οκτωβρίου 2015
Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν είτε σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), είτε στους βοηθούς του μαθήματος στο εργαστήριο των μεταπτυχιακών, την Δευτέρα, 26 Οκτωβρίου 2015, ώρα 15:00-16:00. Ασκήσεις που παραδίδονται μετά τις 16:00 της Δευτέρας, 26/10/2015 θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin. Εναλλακτικά, εκπρόθεσμες ασκήσεις μπορούν να παραδοθούν στη διδάσκουσα ή να σταλούν στο hy240a@csd.uoc.gr μέσω ηλεκτρονικού ταχυδρομείου.

Άσκηση 1 [50 μονάδες]

α. Εξηγήστε ποια δομή (στοίβα ή ουρά) είναι η πιο κατάλληλη για την επίλυση κάθε ενός από τα προβλήματα που παρουσιάζονται στη συνέχεια: [10%]

1.	Ανάγνωση μιας ακολουθίας χαρακτήρων και τύπωση των χαρακτήρων με αντίστροφη σειρά. Για παράδειγμα, αν η ακολουθία ανάγνωσης είναι η OILKARI θα πρέπει να τυπωθεί η ακολουθία χαρακτήρων IRAKLIO.
2.	Ανάγνωση ενός αλφαριθμητικού s και επαλήθευση αν το αλφαριθμητικό είναι της μορφής s = abab όπου a και b είναι οποιαδήποτε αλφαριθμητικά. Για παράδειγμα, το αλφαριθμητικό ALFABETAALFABETA είναι της μορφής που ζητείται, όπου a = ALPHA και b = BETA
3.	Υλοποίησης της εντολής history στο unix η οποία προκαλεί την εκτύπωση των k τελευταίων εντολών που ζητήθηκε να εκτελεστούν από το φλοιό.
4.	Προσομοίωση της λειτουργίας ενός συστήματος αναμονής πελατών σε μια τράπεζα που χρησιμοποιεί σύστημα απόδοσης προτεραιότητας με σειριακούς αριθμούς
5.	Καταχώρηση ασθενών που καταφθάνουν σε ιατρική κλινική προκειμένου να μπορεί να γίνει η ανάθεση ασθενών σε ιατρούς με δίκαιο τρόπο, δηλαδή με τη σειρά που αυτοί έφθασαν στην κλινική.

Για κάθε ένα από τα προβλήματα που παρουσιάζονται παραπάνω, περιγράψτε πιθανή λύση χρησιμοποιώντας τη δομή που προτείνατε.

β. Έστω ότι μια βιβλιοθήκη σας παρέχει πρόσβαση σε στοίβες και ουρές χαρακτήρων. Η βιβλιοθήκη σας επιτρέπει να ορίσετε μια στοίβα (ή μια ουρά) και να καλέσετε τις 5 βασικές λειτουργίες σε αυτή. Για παράδειγμα, ο ορισμός μιας στοίβας (ή μιας ουράς) γίνεται γράφοντας: Stack S; (αντίστοιχα, Queue Q;). Για τη στοίβα υποστηρίζονται οι εξής λειτουργίες: (1) void MakeEmptyStack(Stack S), (2) boolean IsEmptyStack(Stack S), (3) Type Top(Stack S), (4) Type Pop(Stack S), (5) void Push(Stack S, char x). Αντίστοιχα, για την ουρά υποστηρίζονται οι λειτουργίες: (1) void MakeEmptyQueue(Queue Q), (2) boolean IsEmptyQueue(Queue Q), (3) char Front(Queue Q), (4) char Dequeue(Queue Q), (5) void Enqueue(Queue Q, char x).

Επιλύστε τα προβλήματα που παρουσιάζονται στη συνέχεια:

i. Παρουσιάστε ψευδοκώδικα για μια συνάρτηση η οποία θα παίρνει ως όρισμα μια στοίβα και θα τοποθετεί, με τη σειρά που εμφανίζονται στη στοίβα, τα στοιχεία εκείνα που έχουν κλειδί μεγαλύτερο ή ίσο από μια σταθερά c στις υψηλότερες (topmost) θέσεις της στοίβας, χωρίς να αλλάξει τη σειρά με την οποία εμφανίζονται τα υπόλοιπα στοιχεία. Η σταθερά c θα αποτελεί όρισμα της συνάρτησης που θα κατασκευάσετε. Για παράδειγμα, αν η στοίβα περιέχει τα στοιχεία 5,1,7,9,2,3,6,4,8, με το στοιχείο 8 να είναι το κορυφαίο, και η σταθερά c έχει τιμή ίση με 7, τότε μετά την εκτέλεση της συνάρτησης, η στοίβα θα πρέπει να περιέχει τα στοιχεία 5,1,2,3,6,4,7,9,8 με τη σειρά που αναγράφονται (όπου το 8 είναι και πάλι το κορυφαίο στοιχείο). [20%]

ii. Παρουσιάστε ψευδοκώδικα για μια συνάρτηση η οποία θα βρίσκει το στοιχείο με το μεγαλύτερο κλειδί σε μια ουρά και θα το τοποθετεί πρώτο στην ουρά. Αν υπάρχουν περισσότερα από ένα στοιχεία των οποίων το κλειδί είναι μέγιστο, τότε θα πρέπει να τοποθετηθούν όλα στην αρχή της ουράς (διατηρώντας την αρχική σειρά τους). Η διάταξη των υπολοίπων στοιχείων δεν θα πρέπει να αλλάξει. [20%]

Σημειώσεις: I. Οι αλγόριθμοι και για τα δύο ερωτήματα θα πρέπει να χρησιμοποιούν δομές δεδομένων από τη βιβλιοθήκη που σας παρέχεται, καθώς και ενδεχόμενα κάποιες βοηθητικές μεταβλητές. Δεν επιτρέπεται η χρήση πινάκων ή άλλων δομών δεδομένων που δεν ανήκουν στη βιβλιοθήκη για την αποθήκευση ή την επεξεργασία των αλφαριθμητικών ή των ακολουθιών χαρακτήρων. II. Θεωρήστε πως σας παρέχεται μια βοηθητική ρουτίνα char getchar() για την ανάγνωση αριθμών ή χαρακτήρων.

Άσκηση 2 [20 μονάδες]

Έστω πως μια διπλά-συνδεδεμένη λίστα περιέχει στοιχεία του ακόλουθου τύπου:

```
struct node {
    int pid;
    struct node *prev;
    struct node *next;
}
```

Προσαρμόστε τον αλγόριθμο InsertionSort() ώστε αυτός να ταξινομεί τη διπλά-συνδεδεμένη λίστα βάσει του πεδίου pid των κόμβων.

Άσκηση 3 [30 Μονάδες]

Θεωρήστε τη λειτουργία Difference(S_1, S_2) (διαφορά συνόλων), όπου S_1 και S_2 είναι δύο σύνολα.

1. Παρουσιάστε αλγόριθμο που να υλοποιεί την Difference() δεδομένου ότι τα σύνολα υλοποιούνται με **ταξινομημένες απλά-συνδεδεμένες λίστες.** Ο αλγόριθμος σας θα πρέπει να έχει πολυπλοκότητα $O(n_1+ n_2)$, όπου n_1 και n_2 είναι το πλήθος στοιχείων των συνόλων S_1 και S_2. Εξηγήστε γιατί ο αλγόριθμός σας επιτυγχάνει την πολυπλοκότητα αυτή. [15%]

2. Παρουσιάστε αλγόριθμο που να υλοποιεί την Difference() δεδομένου ότι τα σύνολα υλοποιούνται με **μη-ταξινομημένες απλά-συνδεδεμένες λίστες.** Τι πολυπλοκότητα έχει ο αλγόριθμός σας και γιατί; [15%]

Η Difference() πρέπει να παίρνει ως όρισμα ένα δείκτη στο πρώτο στοιχείο της λίστας που υλοποιεί το S_1 και ένα δείκτη στο πρώτο στοιχείο της λίστας που υλοποιεί το S_2. Θα πρέπει να επιστρέφει ένα δείκτη σε μια νέα λίστα, η οποία θα περιέχει εκείνα τα στοιχεία του S_1 που δεν ανήκουν στο S_2. Για το ερώτημα 1, η νέα λίστα θα πρέπει να είναι ταξινομημένη, ενώ για το ερώτημα 2 η νέα λίστα θα πρέπει να είναι μη-ταξινομημένη. Οι λίστες που αναπαριστούν τα S1 και S2 θα πρέπει να παραμένουν αναλλοίωτες κατά την εκτέλεση των αλγορίθμων.