Εργασία 1η

Τελική Ημερομηνία Υποβολής - 25.11.2020

Απαιτητές Δομές Δεδομένων της Εργασίας

Για την διεκπεραίωση της παρούσας εργασίας είναι απαραίτητη η υλοποίηση των δομών δεδομένων της λίστας, της ουράς fifo, της ουράς lifo ή stack και του δυαδικού δέντρου αναζητήσεως που περιγράφονται παρακάτω.

Οι δομές αυτές αποθηκεύουν ακέραιους τύπου int (ή long int) ή οποιονδήποτε ;άλλο τύπο δεδομένων πιστεύετε ότι σας εξυπηρετεί κατά περίπτωση. Κάθε μία από τις δομές αυτές υλοποιείται από δύο αρχεία, ένα αρχείο με κατάληξη .h και ένα αρχείο με κατάληξη .c. Στα αρχεία με κατάληξη .h περιέχονται ένα ή περισσότερα struct που περιγράφουν τη δομή δεδομένων και τα prototypes των συναρτήσεων που την υλοποιούν και στο αρχείο με κατάληξη .c μόνο οι υλοποιήσεις των αντίστοιχων συναρτήσεων που περιγράφονται στο .h αρχείο.

Τα αρχεία που περιγράφουν κάθε μία από τις παρακάτω δομές θα πρέπει να βρίσκονται μέσα στον κατάλογο των αρχείων που υποβάλλετε, διότι απαιτούνται κατά τη μεταγλώττιση από το σχετικό Makefile. Εάν μία δομή πιστεύετε ότι δεν θα σας χρειαστεί στα κυρίως προγράμματα που θα γράψετε (list1.c, tree1.c, tree2.c, tree3.c), μπορείτε να αφήσετε τα αντίστοιχα αρχεία κενά περιεχομένου.

Μια καλή πρακτική που σας προτείνεται να ακολουθήσετε σε όλα τα .h αρχεία (header files), ώστε να αποφύγετε πολλαπλές ενσωματώσεις του .h αρχείου στον υπό μεταγλώττιση κώδικα είναι να γράψετε τον κώδικα σας μέσα σε μία δήλωση της παρακάτω μορφής

```
#ifndef __HEADER_FILE_H_
#define __HEADER_FILE_H_

/* εισάγετε όλο τον κώδικα του
 * header file εδώ
 */

#endif
```

αντικαθιστώντας το τμήμα `HEADER_FILE` με το όνομα του αρχείου με κατάληξη .h. Στο παρακάτω παράδειγμα, μπορείτε να δείτε πως μπορεί να δηλωθεί το περιεχόμενο του header file της διπλά συνδεδεμένης λίστας.

```
#ifndef __DLIST_H_
#define __DLIST_H_

typedef struct dlist_node {
  struct dlist_node* next;
  struct dlist_node* prev;
  int data;
} dnode_t;

typedef struct dlist {
  struct dlist_node *head, *tail;
  int size;
} dlist_t;

/* εδώ μπορείτε να βάλετε τα prototypes των
συναρτήσεως της λίστας. */

#endif
```

Η διπλά συνδεδεμένη λίστα

Στα αρχεία **`dlist.c, dlist.h`** υλοποιήστε μία διπλά συνδεδεμένη λίστα της επιλογής σας.

Η ουρά FIFO

Στα αρχεία **`fifo.c, fifo.h`** υλοποιήστε μία ουρά first-in-first-out της επιλογής σας.

Η στοίβα STACK

Στα αρχεία **`stack.c, stack.h`** υλοποιήστε μία ουρά last-in-first-out της επιλογής σας.

Το δυαδικό δέντρο αναζητήσεως

Στα αρχεία **`tree.c, tree.h`** υλοποιήστε ένα δυαδικό δέντρο αναζητήσεως της επιλογής σας.

Λίστες 1

Γράψτε το πρόγραμμα `list1.c` το οποίο διαβάζει μία ακολουθία ακεραίων αριθμών από την καθιερωμένη είσοδο, για την οποία δεν γνωρίζουμε εκ των προτέρων το μήκος της και την αποθηκεύει σε μία λίστα της επιλογή σας. Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση του αριθμού μηδέν (το μηδέν δεν αποθηκεύεται στη λίστα). Το πρόγραμμα εντοπίζει τη μέγιστη υπο-ακολουθία μέσα στη λίστα της οποίας το άθροισμα είναι μηδέν (0) και εκτυπώνει τα χαρακτηριστικά της στην καθιερωμένη έξοδο (stdout). Εάν υπάρχουν περισσότερες υπο-ακολουθίες με το ίδιο μήκος επιλέγει την πρώτη που συναντούμε, όπως διατρέχουμε τη σειρά των στοιχείων από την αρχή προς το τέλος.

Κατά την εκτύπωση, εκτυπώνει τα εξής:

- τη συμβολοσειρά `"start: START_POS, stop: END_POS, size: SIZE"` ακολουθούμενη από χαρακτήρα αλλαγής γραμμής, όπου `START_POS` η θέση του πρώτου στοιχείου της υπο-ακολουθίας εντός της αρχικής ακολουθίας (υποθέτοντας ότι το πρώτο στοιχείο της αρχικής ακολουθίας αριθμείται με μηδέν), `END_POS` η θέση του τελευταίου στοιχείου της υπο-ακολουθίας και `SIZE` το μέγεθος της υποακολουθίας.

- τα στοιχεία της υπο-ακολουθίας. Μετά από κάθε στοιχείο ακολουθεί ένας κενός χαρακτήρας και μετά τον τελευταίο κενό χαρακτήρα ακολουθεί χαρακτήρας αλλαγής γραμμής.

Για παράδειγμα, για την παρακάτω ακολουθία ακεραίων

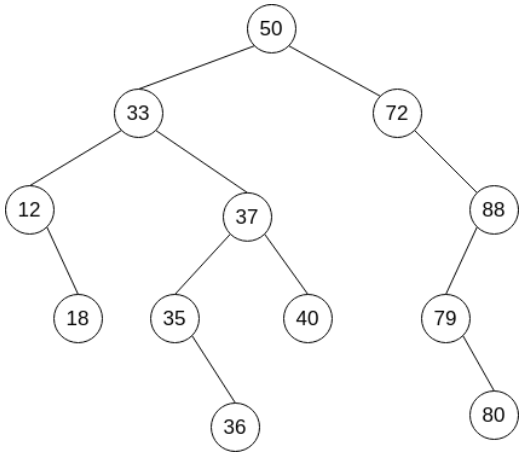
`1 -3 3 3 -4 4 -4 2 2 -3 5`

η μέγιστη υπο-ακολουθία με άθροισμα μηδέν είναι αυτή που ξεκινά από τη θέση 1 και τελειώνει στη θέση 9, ως εξής: **`-3 3 3 -4 4 -4 2 2 -3`**

Δυαδικά Δέντρα 1

Γράψετε το πρόγραμμα `tree1.c`, το οποίο διαβάζει μία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*), η οποία αποτελεί την *postorder* διαπέραση ενός δυαδικού δέντρου αναζητήσεως. Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση ενός αρνητικού αριθμού (ο αρνητικός αριθμός δεν καταχωρείται). Το πρόγραμμα κατασκευάζει το δυαδικό δέντρο αναζητήσεως που αντιστοιχεί στη συγκεκριμένη διαπέραση και εκτυπώνει στο *stdout* την level-order διαπέραση του. Μετά την εκτύπωση κάθε αριθμού εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Για παράδειγμα, για το παρακάτω δυαδικό δέντρο η post-order και η pre-order διαπεράσεις είναι το εξής:



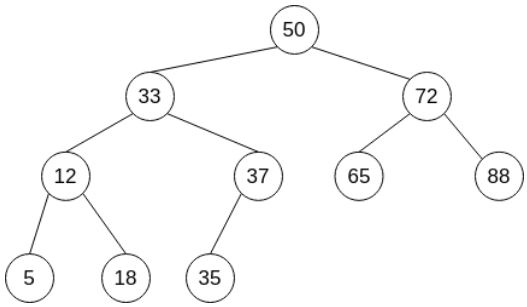

```
PostOrder:
18, 12, 36, 35, 40, 37, 33, 80, 79, 88, 72, 50
```

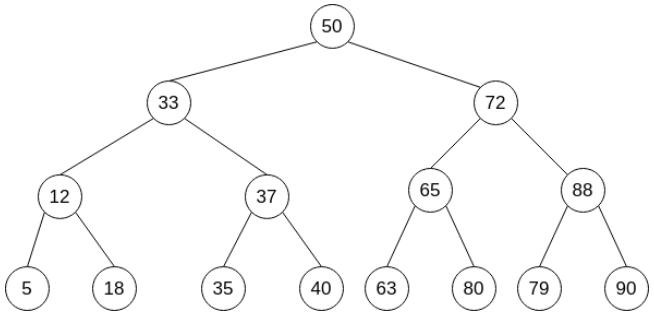
```
Level Order:
50, 33, 72, 12, 37, 88, 18, 35, 40, 79, 36, 80
```

Δυαδικά Δέντρα 2

Γράψετε το πρόγραμμα `tree2.c`, το οποίο διαβάζει μία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*), η οποία αποτελεί την level-order διαπέραση ενός πλήρους δυαδικού δέντρου. Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση ενός αρνητικού αριθμού. Το πρόγραμμά αφού κατασκευάσει το δυαδικό δέντρο ελέγχει εάν το δέντρο αυτό είναι δυαδικό δέντρο αναζητήσεως ή απλό δυαδικό δέντρο και εκτυπώνει αντίστοιχα το μήνυμα `"Binary Search Tree"` ή `"Binary Tree"`, ακολουθούμενο από χαρακτήρα αλλαγής γραμμής.

Υπενθυμίζουμε ότι πλήρες είναι το δυαδικό δέντρο που *α)* όλοι οι εσωτερικοί κόμβοι του έχουν δύο παιδιά (γεμάτο δέντρο), *β)* τα φύλλα βρίσκονται μόνο στα δύο τελευταία επίπεδα και *γ)* τα φύλλα του τελευταίου επιπέδου είναι κατά το δυνατόν πιο αριστερά. Παραδείγματα πλήρους δέντρου δίνονται παρακάτω:





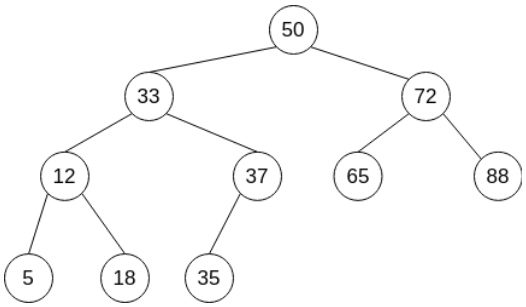
Παρατήρηση: το αριστερό εκ των δύο δέντρων είναι δυαδικό δέντρο αναζητήσεως σε αντίθεση με το δεξί που δεν είναι.

Δυαδικά Δέντρα 3

Γράψετε το πρόγραμμα `tree3.c`, το οποίο διαβάζει μία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*), η οποία αποτελεί την pre-order διαπέραση ενός δυαδικού δέντρου αναζητήσεως. Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση ενός αρνητικού αριθμού. Το πρόγραμμα κατασκευάζει το δυαδικό δέντρο αναζητήσεως που αντιστοιχεί στη συγκεκριμένη διαπέραση.

Στη συνέχεια το πρόγραμμα εκτυπώνει το μήνυμα `"Enter 2 integers: "` και διαβάζει δύο ακέραιους αριθμούς που χωρίζονται μεταξύ τους με κενό χαρακτήρα. Κάθε ακέραιο που διαβάζει, τον αναζητά μέσα στο δέντρο. Εάν ένας αριθμός εκ των δύο ακεραίων δεν υπάρχει στο δέντρο εκτυπώνεται το μήνυμα `"X not found!"` ακολουθούμενο από χαρακτήρα αλλαγής γραμμής και το πρόγραμμα τερματίζει, όπου X ο πρώτος από τους δύο αριθμούς που δεν βρέθηκε. Διαφορετικά εκτυπώνει ~~χαρακτήρα αλλαγής γραμμής και~~ το μήνυμα `"Path is: "`. Στη συνέχεια εκτυπώνεται το ελάχιστο μονοπάτι από τον κόμβο που αντιστοιχεί στην τιμή που διαβάστηκε πρώτη, έως τον κόμβο που αντιστοιχεί στην τιμή που διαβάστηκε δεύτερη. Μετά την εκτύπωση κάθε αριθμού εκτυπώνεται κενός χαρακτήρας, ενώ μετά τον τελευταίο αριθμό ~~και μετά το τελευταίο κενό~~ εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Για παράδειγμα, στο διπλανό δέντρο αναζητήσεως το οποίο δίνεται από την pre-order διαπέραση `50, 33, 12, 5, 18, 37,35, 72, 65, 88` το συντομότερο μονοπάτι από τον κόμβο 35 έως τον κόμβο 88 είναι `35, 37, 33, 50, 72, 88`.



Τρόπος Αποστολής

Η αποστολή της εργασία θα γίνει μέσω της πλατφόρμας autolab (δεν απαιτείται συνδεση VPN). Ακολουθήστε τα εξής βήματα:

1. Φτιάξτε ένα φάκελο με όνομα **`hw1submit`** και αντιγράψτε μέσα εκεί τα αρχεία **dlist.c, dlist.h, stack.c, stack.h, fifo.c, fifo.h, tree.c, tree.h, list1.c, tree1.c, tree2.c, tree3.c**.
2. Συμπιέστε ως tar.gz. Σε Linux/KDE πάτε πάνω στο φάκελο, κάνετε δεξί κλικ και επιλέγετε **Compress -> Here (as tar.gz).** Δημιουργείται το αρχείο **hw1submit.tar.gz**.

3. Συνδέεστε στη διεύθυνση **https://autolab.e-ce.uth.gr** και επιλέγετε το μάθημα **ECE215-F20 (f20)** και από αυτό την εργασία **HW1**.
4. Για να υποβάλετε την εργασία σας κάνετε click στην επιλογή **"I affirm that I have compiled with this course academic integrity policy…"** και πατάτε **submit**. Στη συνέχεια επιλέγετε το αρχείο **hw1submit.tar.g**z που δημιουργήσατε στο βήμα 2.