Εργασία 2η

Τελική Ημερομηνία Υποβολής - 16.12.2020

Προσοχή: Μην αντιγράφετε τα μηνύματα εξόδου του προγράμματος από την εκφώνηση της εργασίας. Σε αυτές τις περιπτώσεις εισάγονται χαρακτήρες που δεν είναι ASCII, με αποτέλεσμα το script που εκτελείται στο autolab να τερματίζεται αναπάντεχα και χωρίς επαρκή αιτιολόγηση και να μην βγάζει τις πραγματικές διαφορές μεταξύ της δικής σας εξόδου και της επιθυμητής.

Απαγορεύεται η χρήση καθολικών μεταβλητών και η χρήση βοηθητικών πινάκων για την εκτέλεση των αλγορίθμων ταξινόμησης.

Ο κώδικας που υποβάλλετε πρέπει να είναι πρωτότυπος και να μην έχει σημαντική ομοιότητα με κώδικα συμφοιτητών σας.

Απαιτητές Δομές Δεδομένων και Αλγόριθμοι της Εργασίας

Για την διεκπεραίωση της παρούσας εργασίας είναι απαραίτητη η υλοποίηση των δομών δεδομένων που περιγράφονται παρακάτω. Οι δομές αυτές αποθηκεύουν ακέραιους τύπου int (ή long int) ή οποιονδήποτε τύπο δεδομένων πιστεύετε ότι σας εξυπηρετεί. Κάθε δομή υλοποιείται από δύο αρχεία, ένα αρχείο με κατάληξη .h και ένα αρχείο με κατάληξη .c. Στα αρχεία με κατάληξη .h περιέχονται τα struct της κάθε δομής δεδομένων και τα prototypes των συναρτήσεων που την υλοποιούν και στο .c μόνο οι υλοποιήσεις των αντίστοιχων συναρτήσεων. Σας παρέχεται κατάλληλο Makefile το οποίο χρησιμοποιείται για την μεταγλώττιση του κώδικα στο autolab.

Η διπλά συνδεδεμένη λίστα

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

Το δυαδικό δέντρο αναζητήσεως

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

Οι αλγόριθμοι ταξινόμησης

Οι υλοποιήσεις των αλγορίθμων ταξινόμησης που ζητούνται στην παρούσα εργασία θα πρέπει να βρίσκονται στο αρχείο **`sort.c`** και τα prototypes των αλγορίθμων αυτών στο αρχείο **`sort.h`**.

Ταξινόμηση

Γράψτε το πρόγραμμα `sort1.c` το οποίο λαμβάνει ένα ή δύο ορίσματα από τη γραμμή εντολών, διαβάζει μία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*) και την αποθηκεύει σε μία διπλά συνδεδεμένη λίστα με τη σειρά που διάβασε. Στη συνέχεια κάνει τα εξής:

1. Εκτυπώνει τα περιεχόμενα της λίστας. Μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
2. Εκτυπώνει ξανά χαρακτήρα αλλαγής γραμμής.
3. Ταξινομεί τους ακέραιους με τη βοήθεια των παρακάτω αλγορίθμων. Οι αλγόριθμοι εφαρμόζονται υποχρεωτικά πάνω σε λίστα. Η επιλογή του αλγορίθμου ταξινόμησης γίνεται με βάση το πρώτο όρισμα της γραμμής εντολών, το οποίο είναι ακέραιος στο εύρος 1 έως 6 ως εξής:
    - 1 -> insertion sort
    - 2 -> selection sort
    - 3 -> quick sort, ως σημείο διαχωρισμού επιλέγεται πάντα το δεξιότερο στοιχείο και εκ των δύο μερών ταξινομείται πρώτο πάντα το αριστερό.
    - 4 -> merge sort
    - 5 -> radix sort lsd
    - 6 -> radix sort msd

    Εάν το πρώτο όρισμα δεν είναι ακέραιος στο εύρος 1 έως 6 εκτυπώνει το μήνυμα "`Invalid argument`" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής και το πρόγραμμα τερματίζει.

    Εάν η επιλογή είναι 5 ή 6, το πρόγραμμα λαμβάνει υποχρεωτικά ένα επιπλέον όρισμα που προσδιορίζει πόσα bits συγκροτούν κάθε λέξη που απαρτίζει τον ακέραιο (δες τη σημείωση παρακάτω). Επιτρεπτές τιμές είναι 1,2,4,8 και 16 bits. Εάν δεν δοθεί δεύτερο όρισμα ή η τιμή που

δίνεται δεν ανήκει στις επιτρεπτές τιμές το πρόγραμμα τερματίζει εκτυπώνοντας το μήνυμα "`Invalid argument`" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής.

4. Για κάθε μία από τις επιλογές 1 έως 6 το πρόγραμμα εκτυπώνει πληροφορία η οποία βρίσκεται στην υποενότητα ενδιάμεσες εκτυπώσεις αλγορίθμων.
5. Εκτυπώνει χαρακτήρα αλλαγής γραμμής και τα περιεχόμενα της ταξινομημένης λίστας. Μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Σημείωση Merge Sort: Για τον αλγόριθμο merge sort μπορείτε να εφαρμόσετε την εξής παραλλαγή για λίστες:

1. Σπάμε αναδρομικά τη λίστα στη μέση σε δύο διακριτές λίστες, την αριστερή και την δεξιά και εφαρμόζουμε αναδρομικά τον αλγόριθμο merge sort για τις λίστες αυτές. Εάν μία εκ των δύο λιστών έχει ένα στοιχείο τότε είναι αυτόματα ταξινομημένη, ενώ εάν έχει δύο στοιχεία την ταξινομούμε συγκρίνοντας τα δύο στοιχεία μεταξύ τους και σε περίπτωση που είναι ανάποδα ταξινομημένα τα αντιμεταθέτουμε.
2. Αφού καλέσουμε τον αλγόριθμο merge sort για τις δύο λίστες, αυτές πλέον έχουν αναδρομικά ταξιονομηθεί.
3. Στο τέλος, ενώνουμε τις δύο αναδρομικά ταξινομημένες λίστες, με τρόπο ώστε η τελική λίστα που θα προκύψει από τις δύο να έχει τα στοιχεία και των δύο λιστών ορθά ταξινομημένα. Επιλέγουμε να τοποθετήσουμε τα στοιχεία στη μία εκ των δύο λιστών (δεν φτιάχνουμε νέα λίστα) και καταστρέφουμε την άλλη.

Σημείωση Radix Sort: Για τους αλγορίθμους Radix Sort LSD και Radix Sort MSD μπορούμε να θεωρήσουμε κάθε ακέραιο αριθμό ως μία λέξη που αποτελείται από γράμματα (ψηφία) μεγέθους X bits έκαστο. Το X μπορεί να πάρει τιμές 1,2,4,8 ή 16. Στη συνέχεια, μπορούμε να ταξινομήσουμε οποιαδήποτε ακολουθία μη προσημασμένων ακεραίων αριθμών με τη βοήθεια των παραπάνω δύο αλγορίθμων κατανομής.

Για παράδειγμα, η δυαδική αναπαράσταση του αριθμού `123456789` είναι `0b111010110111100110100010101`. Ο παρακάτω πίνακας αποτυπώνει τις διαφορετικες επιλογές τις οποίες έχουμε για να χωρίσουμε τον ακέραιο-λέξη σε ψηφία. Στην πρώτη γραμμή του πίνακα εμφανίζονται οι θέσεις των ψηφίων. Ο ακέραιος μπορεί να αναπαρασταθεί ως α) μία λέξη από 32 ψηφία μεγέθους 1 bit έκαστο και αλφάβητο μεγέθους 2 ψηφίων (γραμμή a του πίνακα) , β) μία λέξη από 16 ψηφία μεγέθους 2 bits έκαστο και αλφάβητο μεγέθους 4 (γραμμή b του πίνακα), γ) μία λέξη από 8 ψηφία 4 bits έκαστο και αλφάβητο μεγέθους 16 ψηφίων (γραμμή c), δ) μία λέξη από 4 ψηφία 8 bits έκαστο και αλφάβητο μεγέθους 256 ψηφίων (γραμμή d) ή ε) μία λέξη από 2 ψηφία 16 bits έκαστο και αλφάβητο μεγέθους 2^{16} ψηφίων (τελευταία γραμμή e).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1
b	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1
c	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1
d	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1
e	0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

Ενδιάμεσες εκτυπώσεις αλγορίθμων

Κάθε αλγόριθμος εκτυπώνει έναν αριθμό μηνυμάτων που αποτυπώνουν την πρόοδο εκτέλεσης του αλγορίθμου. Τα μηνύματα αυτά παρουσιάζονται αναλυτικά στην παρούσα υποενότητα.

Insertion Sort & Selection Sort

Το πρόγραμμα εκτυπώνει σε κάθε βήμα (διαπέραση) του αλγορίθμου το αλφαριθμητικό "`[S]:`", όπου S το βήμα του αλγορίθμου με εύρος τιμών από 1 έως και N-1, όπου N ο αριθμός των στοιχείων προς ταξινόμηση. Στη συνέχεια εκτυπώνει τα περιεχόμενα της λίστας στο τρέχον βήμα. Κατά την εκτύπωση της λίστας, μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Παράδειγμα Insertion Sort

Για την ταξινόμηση των στοιχείων **93 13 85 75 71 35 60 44 25 55** ο αλγόριθμος εκτυπώνει τα εξής:

```
[1]: 13 93 85 75 71 35 60 44 25 55
[2]: 13 85 93 75 71 35 60 44 25 55
[3]: 13 75 85 93 71 35 60 44 25 55
[4]: 13 71 75 85 93 35 60 44 25 55
[5]: 13 35 71 75 85 93 60 44 25 55
[6]: 13 35 60 71 75 85 93 44 25 55
[7]: 13 35 44 60 71 75 85 93 25 55
[8]: 13 25 35 44 60 71 75 85 93 55
[9]: 13 25 35 44 55 60 71 75 85 93
```

Παράδειγμα Selection Sort

Για την ταξινόμηση των στοιχείων **93 13 85 75 71 35 60 44 25 55** ο αλγόριθμος εκτυπώνει τα εξής:

```
[1]: 13 93 85 75 71 35 60 44 25 55
[2]: 13 25 85 75 71 35 60 44 93 55
[3]: 13 25 35 75 71 85 60 44 93 55
[4]: 13 25 35 44 71 85 60 75 93 55
[5]: 13 25 35 44 55 85 60 75 93 71
[6]: 13 25 35 44 55 60 85 75 93 71
[7]: 13 25 35 44 55 60 71 75 93 85
[8]: 13 25 35 44 55 60 71 75 93 85
[9]: 13 25 35 44 55 60 71 75 85 93
```

Quick Sort

Το πρόγραμμα εκτυπώνει σε κάθε αναδρομική κλήση του αλγορίθμου τα εξής:

- Κατά την είσοδο στην αναδρομική κλήση
  - εκτυπώνει το αλφαριθμητικό "`==`", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος quicksort. Κατά την αρχική κλήση του quicksort θεωρούμε ότι είμαστε στο επίπεδο 0, οπότε δεν εκτυπώνεται το συγκεκριμένο αλφαριθμητικό.
  - εκτυπώνει το αλφαριθμητικό "`[D >]`", όπου D τρέχον το επίπεδο της αναδρομής. Ακολουθεί το τμήμα της λίστας που πρόκειται να ταξινομηθεί στη συγκεκριμένη κλήση. Δες σημείωση εκτύπωσης παρακάτω.
- Μετα την τοποθέτηση του σημείου διαχωρισμού στην τελική του θέση και <u>ΜΟΝΟ εφόσον το τμήμα προς ταξινόμηση έχει μέγεθος μεγαλύτερο από δύο στοιχεία</u>
  - εκτυπώνει το αλφαριθμητικό "`==`", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος quicksort. Κατά την αρχική κλήση του quicksort θεωρούμε ότι είμαστε στο επίπεδο 0, οπότε δεν εκτυπώνεται το συγκεκριμένο αλφαριθμητικό.

  - εκτυπώνει το αλφαριθμητικό "`[D -]`", όπου D το τρέχον επίπεδο της αναδρομής. Ακολουθεί το τμήμα της λίστας για το οποίο το σημείο διαχωρισμού πήρε την τελική του θέση. Δες σημείωση εκτύπωσης παρακάτω.
- Πριν την έξοδο από την αναδρομική κλήση
  - εκτυπώνει το αλφαριθμητικό "`==`", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος quicksort. Κατά την αρχική κλήση του quicksort θεωρούμε ότι είμαστε στο επίπεδο 0, οπότε δεν εκτυπώνεται το συγκεκριμένο αλφαριθμητικό.
  - εκτυπώνει το αλφαριθμητικό "`[D <]`", όπου D το τρέχον επίπεδο της αναδρομής. Ακολουθεί το ταξινομημένο τμήμα της λίστας. Δες σημείωση εκτύπωσης παρακάτω.

Σημείωση εκτύπωσης: Κατά την εκτύπωση της λίστας ή τμήματος της λίστας, μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Παράδειγμα Quick Sort

Για την ταξινόμηση των στοιχείων **93 13 85 75 71 35 60 44 25 55** ο αλγόριθμος εκτυπώνει τα εξής:

```
[0 >] 93 13 85 75 71 35 60 44 25 55
[0 -] 25 13 44 35 55 75 60 85 93 71
== [1 >] 25 13 44 35
== [1 -] 25 13 35 44
== == [2 >] 25 13
== == [2 <] 13 25
== [1 <] 13 25 35 44
== [1 >] 75 60 85 93 71
== [1 -] 60 71 85 93 75
== == [2 >] 85 93 75
== == [2 -] 75 93 85
== == == [3 >] 93 85
== == == [3 <] 85 93
== == [2 <] 75 85 93
== [1 <] 60 71 75 85 93
[0 <] 13 25 35 44 55 60 71 75 85 93
```

Merge Sort

Το πρόγραμμα εκτυπώνει σε κάθε αναδρομική κλήση του αλγορίθμου τα εξής:

- Κατά την είσοδο στη αναδρομική κλήση
  - εκτυπώνει το αλφαριθμητικό "`==`", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος mergesort. Κατά την αρχική κλήση του mergesort θεωρούμε ότι είμαστε στο επίπεδο 0, οπότε δεν εκτυπώνεται το συγκεκριμένο αλφαριθμητικό.
  - εκτυπώνει το αλφαριθμητικό "`[S >]`", όπου S το τρέχον επίπεδο της αναδρομής. Ακολουθεί το αταξινόμητο τμήμα της λίστας. Το μήνυμα εκτυπώνεται ακόμη και εάν η λίστα έχει ένα στοιχείο, οπότε είναι αυτόματα ταξινομημένη. Δες σημείωση εκτύπωσης παρακάτω.
- Πριν την έξοδο από την αναδρομική κλήση
  - εκτυπώνει το αλφαριθμητικό "`==`", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος mergesort. Όμοια με προηγούμενα, κατά την αρχική κλήση του mergesort θεωρούμε ότι είμαστε στο επίπεδο 0, οπότε δεν εκτυπώνεται το συγκεκριμένο αλφαριθμητικό.
  - εκτυπώνει το αλφαριθμητικό "`[S <]`", όπου S το τρέχον επίπεδο της αναδρομής. Ακολουθεί το ταξινομημένο τμήμα της λίστας. Δες σημείωση εκτύπωσης παρακάτω.

Σημείωση εκτύπωσης: Κατά την εκτύπωση της λίστας ή τμήματος της λίστας, μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Παράδειγμα Merge Sort

Για την ταξινόμηση των στοιχείων **93 13 85 75 71 35 60 44 25 55** ο αλγόριθμος εκτυπώνει τα εξής:

```
[0 >] 93 13 85 75 71 35 60 44 25 55
== [1 >] 93 13 85 75 71
== == [2 >] 93 13
== == [2 <] 13 93
== == [2 >] 85 75 71
== == == [3 >] 85
== == == [3 <] 85
== == == [3 >] 75 71
== == == [3 <] 71 75
== == [2 <] 71 75 85
== [1 <] 13 71 75 85 93
== [1 >] 35 60 44 25 55
== == [2 >] 35 60
== == [2 <] 35 60
== == [2 >] 44 25 55
== == == [3 >] 44
== == == [3 <] 44
== == == [3 >] 25 55
== == == [3 <] 25 55
== == [2 <] 25 44 55
== [1 <] 25 35 44 55 60
[0 <] 13 25 35 44 55 60 71 75 85 93
```

Radix Sort MSD

Το πρόγραμμα αφού χωρίσει τις λέξεις σε επιμέρους λίστες (buckets) για κάθε αναδρομική κλήση του αλγορίθμου εκτυπώνει για κάθε ένα bucket που έχει τουλάχιστον ένα στοιχείο (μη κενό) τα εξής:

- το αλφαριθμητικό "`== `", όσες φορές έχει κληθεί αναδρομικά ο αλγόριθμος radix sort MSD
- το αλφαριθμητικό "`[D, B] (XXXX)`", όπου D το βάθος της αναδρομής, B ο αριθμός του bucket, στον οποίο έχουν αποθηκευτεί ένα ή περισσότερα στοιχεία με τη μορφή λίστας και XXXX η δυαδική αναπαράσταση του αριθμού B. Ο αριθμός του bucket προκύπτει με βάση την αρίθμηση του υπό εξέταση ψηφίου στο τρέχον βήμα με βάση το λεξικό των ψηφίων.
- τα στοιχεία που περιέχονται στο παραπάνω bucket μέσα σε μία λίστα. Κατά την εκτύπωση της λίστας, μετά από κάθε ακέραιο εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Παράδειγμα Radix Sort MSD

Για την ταξινόμηση των στοιχείων **1594525965 602652567 1318344375 805345487 805345485 272766820 1859915130 1025840939 128870542 534180483** με μήκος λέξης 4 bits ο αλγόριθμος εκτυπώνει τα εξής:

```
== [1, 0] (0000) 128870542
== [1, 1] (0001) 272766820 534180483
== == [2, 0] (0000) 272766820
== == [2, 15] (1111) 534180483
== [1, 2] (0010) 602652567
== [1, 3] (0011) 805345487 805345485 1025840939
```

```
== == [2, 0] (0000) 805345487 805345485
== == == [3, 0] (0000) 805345487 805345485
== == == == [4, 0] (0000) 805345487 805345485
== == == == == [5, 9] (1001) 805345487 805345485
== == == == == == [6, 8] (1000) 805345487 805345485
== == == == == == == [7, 12] (1100) 805345487 805345485
== == == == == == == == [8, 13] (1101) 805345485
== == == == == == == == [8, 15] (1111) 805345487
== == [2, 13] (1101) 1025840939
== [1, 4] (0100) 1318344375
== [1, 5] (0101) 1594525965
== [1, 6] (0110) 1859915130
```

Radix Sort LSD

Το πρόγραμμα σε κάθε βήμα του αλγορίθμου αφού ταξινομήσει τις λέξεις με βάση το τρέχον ψηφίο και ενώσει τις επιμέρους λίστες σε μία, εκτυπώνει τα περιεχόμενα της λίστας αυτής ως εξής:

- εκτυπώνει το αλφαριθμητικό "`[S]`", όπου S το τρέχον βήμα του αλγορίθμου που αντιστοιχεί παράλληλα στην σειρά του ψηφίου που εξετάζεται στο βήμα αυτό (ξεκινάμε από το βήμα με αριθμό 0).
- εκτυπώνει τα περιεχόμενα της λίστας με τη σειρά που αυτά εμφανίζονται. Δίπλα σε κάθε αριθμό μέσα σε παρένθεση εμφανίζεται σε δυαδική μορφή το ψηφίο του αριθμού που ταξινομήθηκε στο τρέχον βήμα.

Παράδειγμα Radix Sort MSD

Για την ταξινόμηση των στοιχείων **1594525965 602652567 1318344375 805345487 805345485 272766820 1859915130 1025840939 128870542 534180483** με μήκος λέξης 4 bits ο αλγόριθμος εκτυπώνει τα εξής:

```
 [0] 534180483(0011) 272766820(0100) 602652567(0111) 1318344375(0111) 1859915130(1010)
1025840939(1011) 1594525965(1101) 805345485(1101) 128870542(1110) 805345487(1111)
 [1] 1594525965(0000) 1025840939(0010) 272766820(0110) 1859915130(0111) 534180483(1000)
128870542(1000) 602652567(1001) 1318344375(1011) 805345485(1100) 805345487(1100)
 [2] 534180483(0010) 1318344375(0110) 1025840939(0111) 272766820(0111) 128870542(1000)
805345485(1000) 805345487(1000) 1594525965(1001) 1859915130(1101) 602652567(1111)
 [3] 1859915130(0000) 1025840939(0001) 272766820(0001) 1318344375(0101) 128870542(0110)
1594525965(1000) 805345485(1001) 805345487(1001) 602652567(1011) 534180483(1111)
 [4] 805345485(0000) 805345487(0000) 272766820(0010) 1318344375(0100) 1025840939(0101)
534180483(0110) 1594525965(1010) 602652567(1011) 1859915130(1100) 128870542(1110)
 [5] 805345485(0000) 805345487(0000) 1594525965(0000) 1025840939(0010) 272766820(0100)
1318344375(1001) 128870542(1010) 534180483(1101) 1859915130(1101) 602652567(1110)
 [6] 805345485(0000) 805345487(0000) 272766820(0000) 602652567(0011) 128870542(0111)
1025840939(1101) 1318344375(1110) 1859915130(1110) 1594525965(1111) 534180483(1111)
 [7] 128870542(0000) 272766820(0001) 534180483(0001) 602652567(0010) 805345485(0011)
805345487(0011) 1025840939(0011) 1318344375(0100) 1594525965(0101) 1859915130(0110)
```

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 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 στο *stdout* το μήνυμα "`Enter integer:` " και διαβάζει έναν ακέραιο από το πληκτρολόγιο τον οποίο αναζητεί μέσα στο δέντρο. Εάν ο αριθμός δεν υπάρχει εκτυπώνει το μήνυμα "`W not found!`" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής και το πρόγραμμα τερματίζει, όπου W ο ακέραιος που διαβάστηκε. Διαφορετικά εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "`Integers in level P are:` " , όπου P το επίπεδο στο οποίο βρέθηκε ο ακέραιος που δόθηκε παραπάνω (θεωρούμε ως 0 το επίπεδο της ρίζας). Στη συνέχεια εκτυπώνει στο *stdout* τους υπόλοιπους αριθμούς που βρίσκονται στο ίδιο επίπεδο με τον αριθμό που διαβάστηκε, εκτυπωμένους σε αύξουσα σειρά. Μετά την εκτύπωση κάθε αριθμού εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

Δυαδικά Δέντρα 2

Γράψετε το πρόγραμμα `tree2.c`, το οποίο διαβάζει μία τυχαία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*). Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση ενός αρνητικού αριθμού. Η ακολουθία ΔΕΝ αντιπροσωπεύει την διαπέραση οποιουδήποτε δυαδικού δέντρου.

Το πρόγραμμα κατασκευάζει από τα στοιχεία της ακολουθίας ένα δυαδικό δέντρο αναζητήσεως το οποίο περιέχει όλα τα στοιχεία της ακολουθίας και στο οποίο η απόσταση της ρίζας από οποιοδήποτε φύλλο του είναι ίση με το ακέραιο μέρος του αριθμού $\log_2 N$ (δηλ. $\lfloor \log_2 N \rfloor$ ή `floor(log2N)`), όπου N ο αριθμός των κόμβων του δέντρου.

Στη συνέχεια το πρόγραμμα εκτυπώνει στο *stdout* την pre-order διαπέραση του δέντρου που κατασκεύασε. Μετά την εκτύπωση κάθε αριθμού εκτυπώνεται κενός χαρακτήρας και μετά το τελευταίο κενό εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

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 **`hw2submit`** και αντιγράψτε μέσα εκεί τα αρχεία **dlist.c, dlist.h, tree.c, tree.h, sort.c, sort.h, sort1.c, tree1.c, tree2.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 **hw2submit.tar.gz**.
3. Συνδέεστε στη διεύθυνση **https://autolab.e-ce.uth.gr** και επιλέγετε το μάθημα **ECE215-F20 (f20)** και από αυτό την εργασία **HW2**.
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.