Ύλη Εργαστηρίου

- Σειριακή και Δυαδική Αναζήτηση
- Αναδρομικές Συναρτήσεις
- Δομή πίνακα - Βασικές λειτουργίες
- Λίστες (Απλή, Διπλή, κυκλική)
- Δομή Στοίβας
- Δομής Ουράς
- Αλγόριθμοι Ταξινόμησης
  - Ταξινόμηση Παρεμβολής (Insertion Sort)
  - Ταξινόμηση Επιλογής (Selection Sort)
  - Ταξινόμηση Φυσαλίδας (Bubble Sort)
  - Ταξινόμηση Συγχώνευσης (Merge Sort)
  - Γρήγορη Ταξινόμηση (Quick Sort)
  - Ταξινόμηση Σωρού (Heap Sort)
- Δυαδικά Δέντρα Αναζήτησης
- Ουρές Προτεραιότητας (Heap)
- Κατακερματισμός

Προηγούμενα Θέματα Εξετάσεων - 1ο θέμα

Έστω μια στοίβα **MyStack** που περιέχει ακέραιους αριθμούς.

Χρησιμοποιώντας όποιες από τις ακόλουθες συναρτήσεις επιθυμείτε : **int pop()**, **void push(int b)**, **int isEmpty()**, γράψτε κώδικα σε C/C++ ο οποίος θα διαγράφει το στοιχείο της στοίβας που βρίσκετε στο τέλος (μηδενικό στοιχείο του πίνακα).

Μπορείτε να χρησιμοποιήσετε και μια δεύτερη βοηθητική στοίβα. Δεν θα πρέπει να υλοποιήσετε τις προαναφερόμενες συναρτήσεις.

top	**5**
	4
	2
	9
0	**7**

top	**5**
	4
	2
0	**9**

Προηγούμενα Θέματα Εξετάσεων - 2ο θέμα

Υλοποιήσετε συνάρτηση με υπογραφή ***int IsCorrect(char str[])***. Η συνάρτηση αυτή θα δέχεται ως παράμετρο μία συμβολοσειρά η οποία θα αποτελείται μόνο από τους χαρακτήρες : '[', ']' , '(', ')', '{' και '}' (όχι απαραίτητα όλους).

Η συνάρτηση θα επιστρέφει 1 αν η συμβολοσειρά περιέχει ακολουθία στην οποία οι παρενθέσεις, οι αγκύλες και τα άγκιστρα χρησιμοποιούνται με ορθό τρόπο δηλαδή ανοίγουν και κλείνουν με τον ενδεδειγμένο τρόπο και θα επιστρέφει -1 σε αντίθετη περίπτωση.

Έτσι για παράδειγμα αν καλέσουμε τη συνάρτηση με παράμετρο την συμβολοσειρά "([])" θα πρέπει να επιστρέψει -1 ενώ αν καλέσουμε τη συνάρτηση με την συμβολοσειρά "([{}])()" θα πρέπει να επιστρέψει 1.

Για να υλοποιήσετε τη συνάρτηση αυτή πρέπει να αποφασίσετε ποια είναι η πιο κατάλληλη δομή δεδομένων και να υλοποιήσετε τις απαραίτητες λειτουργίες της.

Προηγούμενα Θέματα Εξετάσεων - 3ο θέμα

- Υλοποιήσετε συνάρτηση με υπογραφή ***void ListInsertAfter(struct node *head, int data1, int data2)***. Η συνάρτηση αυτή θα διασχίζει την μονοδεσμική λίστα που περνά ως παράμετρο στη συνάρτηση, και θα βρίσκει την 1η εμφάνιση του κόμβου με τιμή (δεδομένο κόμβου) *data1*.
- Ακολούθως θα δημιουργεί και θα εισάγει έναν νέο κόμβο με τιμή *data2* αμέσως μετά τον κόμβο που βρήκε. Τα υπόλοιπα στοιχεία της λίστας (σε περίπτωση που υπάρχουν) θα πρέπει να παραμένουν στην λίστα αμέσως μετά τον νέο κόμβο χωρίς να αλλάζει με κανέναν τρόπο η σειρά τους.
- Στη περίπτωση που δεν υπάρχει κόμβος με τιμή *data1* δεν θα γίνεται καμία τροποποίηση της αρχικής λίστας.

Προηγούμενα Θέματα Εξετάσεων - 4ο θέμα

Υλοποιήστε δύο από τις ακόλουθες λειτουργίες για μια συνδεδεμένη λίστα (linked list):

α) int GetIndNode(struct node* head, int index)

Η συνάρτηση δέχεται ως παραμέτρους μια συνδεδεμένη λίστα και έναν ακέραιο αριθμό που δηλώνει την θέση ενός κόμβου (0 ο 1ος κόμβος, 1 ο 2ος κλπ) και επιστρέφει την τιμή (ακέραια) που είναι αποθηκευμένη στον συγκεκριμένο κόμβο.

β) void DeleteList(struct node head)**

Η συνάρτηση θα πρέπει να διαγράφει όλα τα στοιχεία της λίστας head και να εκτελεί αποδέσμευση μνήμης όπου απαιτείται.

γ) void InsertIndNode(struct node head, int index, int data)**

Η συνάρτηση θα πρέπει να εισάγει στην λίστα που δέχεται ως παράμετρο, στη θέση index έναν νέο κόμβο με τιμή data.

Σε όλες τις υλοποιήσεις θα πρέπει να γίνονται όλοι οι απαραίτητοι έλεγχοι και να εμφανίζονται κατάλληλα μηνύματα λάθους.

Προηγούμενα Θέματα Εξετάσεων - 5ο θέμα

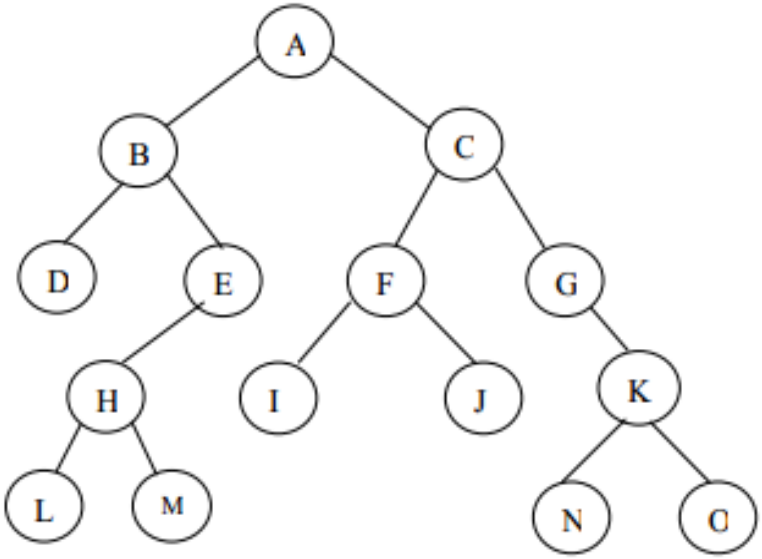
Σε μια δομή δυαδικού δέντρου αναζήτησης εκτελούνται οι παρακάτω πράξεις:

1. InsertNode(15)
2. InsertNode(9)
3. InsertNode(11)
4. InsertNode(17)
5. InsertNode(6)
6. DeleteNode(9)
7. InsertNode(13)
8. InsertNode(12)
9. DeleteNode(11)

Ποιο είναι το δυαδικό δέντρο αναζήτησης που προκύπτει μετά την εισαγωγή/κάθε στοιχείου. Οι πράξεις εκτελούνται με την σειρά που αναφέρονται. Δείξτε σχηματικά πως μεταβάλλεται το δέντρο σε κάθε πράξη.

Προηγούμενα Θέματα Εξετάσεων - 6ο θέμα

Υλοποιήστε συνάρτηση σε C/C++ που θα δέχεται την ρίζα ενός δέντρου και θα υλοποιεί την ενδοδιατεταγμένη διάσχιση (inorder), είτε με επαναληπτική είτε με αναδρομική διαδικασία. Ακολούθως για το παρακάτω δένδρο γράψτε τη σειρά επίσκεψης των κόμβων του χρησιμοποιώντας αυτό τον τρόπο διάσχισης.



Προηγούμενα Θέματα Εξετάσεων - 7ο θέμα

Είναι η ακολουθία [23, 17, 14, 6, 13, 10, 1, 5, 6, 12] ένας σωρός μεγίστου;

Σχεδιάστε το αντίστοιχο δέντρο που αναπαριστά τον σωρό και δικαιολογήστε την απάντηση σας.

Προηγούμενα Θέματα Εξετάσεων - 8ο θέμα

Με δεδομένο τον ακόλουθο πίνακα χαρακτήρων

Κ	Α	Χ	Β	Ω	Ε	Δ	Δ

δείξτε αναλυτικά το πώς μεταβάλλεται ο πίνακας σε κάθε επαναληπτική εκτέλεση του αλγορίθμου ταξινόμησης επιλογής (Selection Sort). Το τελικό αποτέλεσμα του αλγορίθμου θα πρέπει να είναι η αύξουσα ταξινόμηση του πίνακα.

Προηγούμενα Θέματα Εξετάσεων - 9ο θέμα

Περιγράψτε με σύντομο τρόπο ποια μπορεί να είναι η λειτουργία της ακόλουθης συνάρτησης:

```
void myfunc(int n) {
Stack S;


while (n > 0) {
        push(&S, n%2);
        n = n/2;
}

while (!isEmpty(&S))
   printf("%d ", pop(&S));
}
```

Προηγούμενα Θέματα Εξετάσεων - 10ο θέμα

Έστω ότι σε μια απλά συνδεδεμένη λίστα έχουμε στη διάθεση μας δύο δείκτες. Έναν δείκτη για τον 1ο κόμβο και έναν δείκτη για τον τελευταίο. Για ποια/ποιες λειτουργίες η χρονική πολυπλοκότητα εξαρτάται από το μέγεθος της λίστας:

- Εισαγωγή νέου κόμβου στην αρχή της λίστας
- Εισαγωγή νέου κόμβου στο τέλος της λίστας
- Διαγραφή του τελευταίου κόμβου της λίστας
- Διαγραφή του πρώτου κόμβου της λίστας

Προηγούμενα Θέματα Εξετάσεων - 11ο θέμα

Έστω απλά συνδεδεμένη λίστα με δείκτη head στον 1ο κόμβο. Ποια θεωρείτε ότι είναι η λειτουργία της ακόλουθης συνάρτησης:

```
void myfunc(struct node* head){
if (head == NULL)
   return;
  myfunc(head->next);
  printf("%d ", head->value);
}
```

Προηγούμενα Θέματα Εξετάσεων - 12ο θέμα

Ο ακόλουθος κώδικας δέχεται ως παράμετρο μια απλά συνδεδεμένη λίστα και την τροποποιεί μετακινώντας το τελευταίο στοιχείο στην αρχή της λίστας. Επιστρέφει την λίστα τροποποιημένη. Συμπληρώστε το κενό (Σημείο 1) με το κατάλληλο τμήμα κώδικα.

```
Node * myfunc(Node *head) {
 Node *temp, *temp2;
 if ((head == NULL || (head->next == NULL))
   return head;
 temp2 = NULL; temp = head;
 while (temp -> next !=NULL){
   temp2 = temp;
   temp = temp -> next;
 }
 // Σημείο 1
 return head; }
```