Ενδεικτικά θέματα Β' προόδου

Θέμα 1

Γράψτε μια συνάρτηση με όνομα fun() που να δέχεται ως παραμέτρους έναν πίνακα a, το πλήθος των στοιχείων του n, μια θέση στον πίνακα pos και μια τιμή x. Ο πίνακας a και η τιμή x να μπορεί να είναι οποιουδήποτε τύπου (υπόδειξη: χρησιμοποιήστε template). Η συνάρτηση να επιστρέφει τον πίνακα a με την τιμή x να βρίσκεται στη θέση pos και όλα τα στοιχεία του πίνακα που βρίσκονται από τη θέση pos και μετά να έχουν μετακινηθεί μια θέση προς τα δεξιά (η δεξιότερη τιμή του πίνακα χάνεται). Αν το pos έχει τιμή εκτός των ορίων [0,n-1] η συνάρτηση να επιστρέφει χωρίς να αλλάζει τον πίνακα. Γράψτε πρόγραμμα που να καλεί τη συνάρτηση από τη main() για τον πίνακα a={3,7,16,2,8,9}, τη θέση pos=2 και την τιμή x=21.

Θέμα 2

Ο ακόλουθος κώδικας δημιουργεί μια απλά συνδεδεμένη λίστα με 4 κόμβους. Συμπληρώστε τον κώδικα έτσι ώστε στη main να πραγματοποιείται κλήση της συνάρτησης length με όρισμα τη μεταβλητή head και να εμφανίζεται το αποτέλεσμα που επιστρέφει. Να συμπληρωθεί η συνάρτηση length() η οποία θα πρέπει να επιστρέφει το μήκος της συνδεδεμένης λίστας δεδομένου ενός δείκτη που δείχνει στο πρώτο στοιχείο της.

```
#include <iostream>

using namespace std;
struct node {
  int data;
  struct node *next;
};

int length(struct node *node) {









}

int main(void) {
  struct node *node1 = new node{10, NULL};
  struct node *node2 = new node{20, NULL};
  struct node *node3 = new node{30, NULL};
  struct node *node4 = new node{40, NULL};
  struct node *head = node1;
  node1->next=node2;
  node2->next=node3;
  node3->next=node4;

}
```

Θέμα 3

Να δηλωθεί ο κόμβος μιας διπλά συνδεδεμένης λίστας που περιέχει ακέραιες τιμές. Να δημιουργηθεί μια λίστα με τρεις κόμβους. Να δηλωθούν δύο δείκτες head και tail που να δείχνουν ο πρώτος στην αρχή της λίστας και ο δεύτερος στο τέλος της λίστας. Να γραφεί μια συνάρτηση (print) που να διανύει την λίστα από την αρχή προς το τέλος εμφανίζοντας τα στοιχεία της και μια άλλη συνάρτηση (print_reverse) που να διανύει τη λίστα από το τέλος προς την αρχή επίσης εμφανίζοντας τα στοιχεία της λίστας.

Θέμα 4

Δημιουργήστε μια απλά συνδεδεμένη λίστα ακεραίων με 4 στοιχεία. Γράψτε μια συνάρτηση που να δέχεται ως παράμετρο έναν δείκτη προς το πρώτο στοιχείο της λίστας και να επιστρέφει το άθροισμα όλων των στοιχείων της λίστας. Καλέστε τη συνάρτηση έτσι ώστε να εμφανίζει το αποτέλεσμα στη main.

Θέμα 5

Να γράψετε πρόγραμμα που ο χρήστης να εισάγει 5 ακέραιες τιμές που θα δίνει ο χρήστης τόσο σε μια στοίβα όσο και σε μια ουρά. Χρησιμοποιώντας τα δεδομένα που θα βρίσκονται στην στοίβα και στην ουρά υπολογίστε και εμφανίστε το άθροισμα από τις 3 πρώτες και από τις 3 τελευταίες εκ των τιμών που εισήγαγε ο χρήστης.

Θέμα 6

Στο πρόγραμμα stack_oo.cpp του εργαστηρίου 5 προσθέστε μια συνάρτηση με όνομα push_values() που να δέχεται ως παράμετρο ένα διάνυσμα (vector) τιμών και να ωθεί στη στοίβα όλες τις τιμές του διανύσματος εφόσον αυτό είναι δυνατό. Καλέστε τη συνάρτηση push_values από τη main για μια στοίβα με μέγιστη χωρητικότητα 100 θέσεων έτσι ώστε να τοποθετηθούν σε αυτή 5 τιμές τις επιλογής σας.

Θέμα 7

Έστω ότι μια τράπεζα που διαθέτει έναν αριθμό ταμιών για την εξυπηρέτηση των πελατών της. Όταν ένας ταμίας είναι διαθέσιμος εξυπηρετεί τον πελάτη που έχει το εισιτήριο με το μικρότερο χρόνο έκδοσης. Να γράψετε πρόγραμμα που να εμφανίζει το ακόλουθο μενού:

1. Διαθέσιμος ταμίας
2. Νέος πελάτης
3. Έξοδος

Όταν η επιλογή είναι 1 να εμφανίζεται το όνομα του πελάτη που είναι η σειρά του να εξυπηρετηθεί. Όταν η επιλογή είναι 2 να εισάγονται νέα στοιχεία πελάτη: όνομα και χρόνος προσέλευσης (για απλότητα θεωρείστε ότι ο χρόνος προσέλευσης είναι μια ακέραια τιμή στο διάστημα [0,100]). Στην επιλογή 3 το πρόγραμμα να τερματίζει και να εμφανίζεται το πλήθος των πελατών που εξυπηρετήθηκαν και το πλήθος των πελατών που έχουν παραμείνει στην ουρά. Για την υλοποίηση να οριστεί η δομή customer με πεδία name και time και οι πελάτες να τοποθετούνται σε μια ουρά προτεραιότητας ελαχίστων (std::priority_queue).

Θέμα 8

Ένας σωρός ελαχίστων (MINHEAP) έχει αποθηκευτεί σε ένα πίνακα ως εξής: [5,7,8,10,13,15,11,12,19,16].

1. Σχεδιάστε το σωρό στη δεντρική του μορφή. Προσθέστε την τιμή 6 και σχεδιάστε εκ νέου το σωρό ως δένδρο.
2. Στον αρχικό σωρό διαγράψτε την ελάχιστη τιμή και σχεδιάστε εκ νέου το σωρό ως δένδρο. Δείξτε ποια θα είναι η απεικόνιση του σωρού και ως πίνακας.

Θέμα 9

Έστω ένας σωρός μεγίστων 10 στοιχείων με μέγιστη χωρητικότητα 100 θέσεων. Τα στοιχεία του είναι αποθηκευμένα σε ένα μονοδιάστατο πίνακα ως εξής:

NULL	99	64	42	54	32	28	6	19	7	NULL	…	NULL
0	1	2	3	4	5	6	7	8	9	10		100

1. Σχεδιάστε το σωρό στη δενδρική του μορφή.
2. Προσθέστε την τιμή 67 στο σωρό και σχεδιάστε εκ νέου το σωρό.
3. Τροποποιήστε τη συνάρτηση insert_key έτσι ώστε να επιστρέφει το πλήθος των αντιμεταθέσεων που προκαλεί η εισαγωγή ενός νέου στοιχείου key στο σωρό.

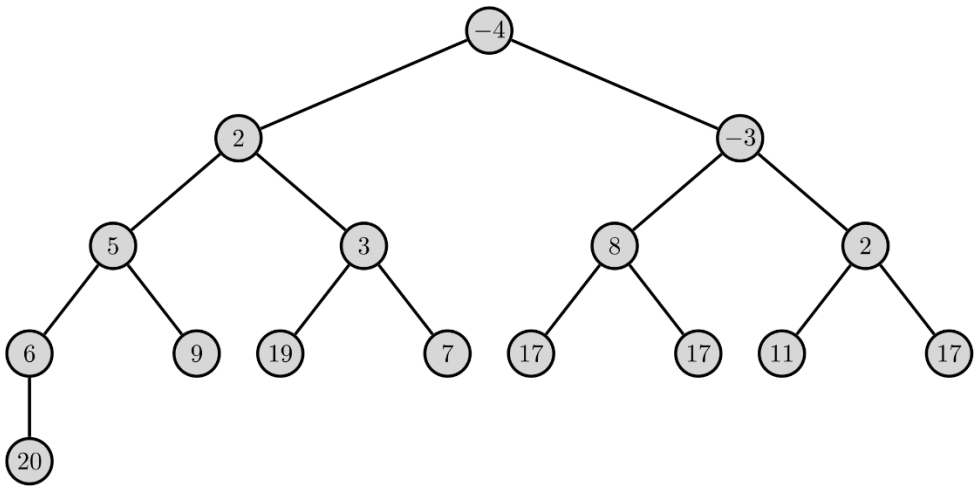
```
int heap[101];
int heap_size = 0;

...

void insert_key(int key)
{
    heap_size++;
    heap[heap_size] = key;
    int pos = heap_size;
    while (pos != 1 && heap[pos / 2] < heap[pos])
    {
        swap(heap[pos / 2], heap[pos]);
        pos = pos / 2;
    }
}
```

Θέμα 10

Έστω ένας σωρός ελαχίστων 10 στοιχείων με μέγιστη χωρητικότητα 100 θέσεων. Η αναπαράσταση του σωρού ως δένδρο παρουσιάζεται στη συνέχεια:



1. Καταγράψτε το σωρό στην αναπαράστασή του ως πίνακα.
2. Διαγράψτε τη μικρότερη τιμή και σχεδιάστε εκ νέου το σωρό ως δένδρο.

Οι λύσεις για τα θέματα 1 έως και 7 βρίσκονται στο https://github.com/chgogos/ceteiep_dsa/tree/master/prep