Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχ. Υπολογιστών
Προγραμματισμός Ηλεκτρονικών Υπολογιστών, 1ο εξάμηνο
https://courses.softlab.ntua.gr/progintro/
***Ε. Ζάχος*, Ν. Παπασπύρου, Δ. Φωτάκης, Π. Ποτίκας, Δ. Σούλιου, Κ. Τζαμαλούκας**

8η Σειρά Ασκήσεων

Οι παρακάτω ασκήσεις προορίζονται για εξάσκηση στη δυναμική διαχείριση μνήμης και τις δυναμικές δομές δεδομένων (π.χ. συνδεδεμένες λίστες, δέντρα). Το αυτόματο σύστημα υποβολής δεν ελέγχει κατά πόσο πραγματικά τις έχετε υλοποιήσει χρησιμοποιώντας δυναμικές δομές δεδομένων. Η υποβολή μπορεί να γίνεται από οποιονδήποτε υπολογιστή, και εκτός εργαστηρίου.

Άσκηση 18.

Έστω ο **αφηρημένος τύπος δεδομένων** `queue` για την παράσταση μιας ουράς ακεραίων αριθμών, ο οποίος υποστηρίζει μόνο τις ακόλουθες λειτουργίες:

```
class queue {
public:
       queue   ();         /* κατασκευαστής: κατασκευάζει μία κενή ουρά */
  bool empty   ();         /* ελέγχει αν η ουρά είναι κενή */
  void enqueue (int x);    /* εισάγει ένα στοιχείο στην ουρά */
  int  dequeue ();         /* αφαιρεί ένα στοιχείο από μια μη κενή ουρά */
  int  peek    ();         /* επιστρέφει (χωρίς να αφαιρεί) το πρώτο στοιχείο μιας μη κενής ουράς */
};
```

Υλοποιήστε αυτόν τον αφηρημένο τύπο δεδομένων χρησιμοποιώντας συνδεδεμένη λίστα.

Στη συνέχεια, χρησιμοποιήστε τον για να γράψετε ένα πρόγραμμα που να διαβάζει μία ακολουθία μη μηδενικών ακέραιων αριθμών (οσοδήποτε πολλών) και να ελέγχει αν αυτή:

1) έχει τόσους θετικούς αριθμούς όσους και αρνητικούς, και (συγχρόνως)
2) οι αρνητικοί αριθμοί εμφανίζονται με την ίδια σειρά κατ' απόλυτο τιμή που εμφανίζονται και οι θετικοί αριθμοί.

Αν αυτό συμβαίνει, το πρόγραμμα πρέπει να εκτυπώνει το μήνυμα "yes", διαφορετικά να εκτυπώνει το μήνυμα "no".

Παραδείγματα εισόδου:

`3 -3 -4 -6 4 -7 6 7`	`3 -3 4 -6 -4 -7 6 7`

Παραδείγματα εξόδου:

`yes`	`no`

► Να υποβληθεί στο αυτόματο σύστημα υποβολής και ελέγχου μέχρι την Κυριακή 17/1/2021

Άσκηση 19.

Έστω ο **αφηρημένος τύπος δεδομένων** `list` για την παράσταση μιας γραμμικής λίστας ακεραίων αριθμών, ο οποίος υποστηρίζει τις ακόλουθες λειτουργίες:

```
class list {
public:
       list   ();              /* κατασκευαστής: κατασκευάζει μία κενή λίστα */
  bool empty  ();              /* ελέγχει αν η λίστα είναι κενή */
  int  size   ();              /* επιστρέφει το μέγεθος της λίστας */
  void add    (int k, int x);  /* εισάγει το στοιχείο x στη θέση k της λίστας */
```

```
    int  get    (int k);          /* επιστρέφει την τιμή του στοιχείου στη θέση k της λίστας */
    void remove (int k);          /* διαγράφει το στοιχείο στη θέση k της λίστας */
};
```

Υλοποιήστε αυτόν τον αφηρημένο τύπο δεδομένων χρησιμοποιώντας συνδεδεμένη λίστα.

Να χρησιμοποιήστε την υλοποίηση του αφηρημένου τύπου δεδομένων για να γράψετε ένα πρόγραμμα που εισάγει και διαγράφει στοιχεία σε μια γραμμική λίστα. Το πρόγραμμά σας θα τυπώνει το μέγεθος της γραμμικής λίστας (έπειτα από την ακολουθία εισαγωγών και διαγραφών) καθώς και την τιμή του στοιχείου σε μία επιλεγμένη θέση της λίστας.

Συγκεκριμένα, το πρόγραμμά σας πρέπει να:

- Διαβάζει από την πρώτη γραμμή της εισόδου το πλήθος των εισαγωγών **N**
- Διαβάζει από καθεμία από τις επόμενες **N** γραμμές της εισόδου δύο θετικούς φυσικούς αριθμούς **K** και **X** και εισάγει τον αριθμό **X** στην **K**-οστή θέση της λίστας.
- Διαβάζει από την **N**+2 γραμμή της εισόδου το πλήθος των διαγραφών **M** < **N**
- Διαβάζει από καθεμία από τις επόμενες **M** γραμμές της εισόδου έναν θετικό φυσικό αριθμό **K** και διαγράφει το στοιχείο στην **K**-οστή θέση της λίστας.
- Διαβάζει από την **N**+**M**+3 γραμμή της εισόδου έναν θετικό φυσικό αριθμό **K** και τυπώνει στην έξοδο δύο φυσικούς αριθμούς (χωρισμένους με ένα κενό) που αντιστοιχούν στο τρέχον μέγεθος της λίστας και στην τιμή του στοιχείου στη θέση **K** της λίστας.

Σε όλες τις παραπάνω περιπτώσεις, να θεωρήσετε ως δεδομένο ότι το **K** θα είναι πάντα μικρότερο ή ίσο του τρέχοντος μεγέθους της λίστας (ή του τρέχοντος μεγέθους της λίστας συν ένα, για τις εισαγωγές).

Παραδείγματα εισόδου:

```
3
1 1
1 2
1 3
1
1
2
```

```
3
1 3
2 2
3 1
1
1
1
```

```
3
1 3
2 2
2 1
1
2
1
```

Παραδείγματα εξόδου:

```
2 1
```

```
2 2
```

```
2 3
```

► Να υποβληθεί στο αυτόματο σύστημα υποβολής και ελέγχου μέχρι την Κυριακή 17/1/2021

Άσκηση 20.

Στην υλοποίηση του αφηρημένου τύπου δεδομένων `list` της προηγούμενης άσκησης, προσθέτουμε την παρακάτω λειτουργία (μέθοδο):

```
int searchMF (int x);
```

Αυτή η λειτουργία επιστρέφει τη θέση της λίστας όπου υπάρχει το στοιχείο `x` (την πρώτη, αν υπάρχουν πολλές θέσεις με στοιχείο το `x`) και μετακινεί το στοιχείο `x` στην πρώτη θέση της λίστας (χωρίς να αλλάξει κάτι άλλο στις σχετικές θέσεις των στοιχείων της λίστας). Αν το στοιχείο `x` δεν υπάρχει στη λίστα, η `searchMF` επιστρέφει την τιμή 0.

Να υλοποιήσετε τη λειτουργία `searchMF`. Στη συνέχεια, να γράψετε ένα πρόγραμμα που εισάγει στοιχεία σε μια γραμμική λίστα και πραγματοποιεί διαδοχικές αναζητήσεις με κλήση της `searchMF`. Το πρόγραμμά σας θα τυπώνει το άθροισμα των τιμών που επιστρέφει η λειτουργία `searchMF` για όλες τις αναζητήσεις.

Συγκεκριμένα, το πρόγραμμά σας πρέπει να:

- Διαβάζει από την πρώτη γραμμή της εισόδου το πλήθος των εισαγωγών **N** για τη δημιουργία της λίστας.

- Διαβάζει από καθεμία από τις επόμενες **N** γραμμές της εισόδου δύο θετικούς φυσικούς αριθμούς **K** και **X** και εισάγει τον αριθμό **X** στην **K**-οστή θέση της λίστας. Να θεωρήσετε ως δεδομένο ότι το **K** θα είναι πάντα μικρότερο ή ίσο του τρέχοντος μεγέθους της λίστας συν ένα.
- Διαβάζει από την **N**+2 γραμμή της εισόδου το πλήθος των αναζητήσεων **M**
- Διαβάζει από καθεμία από τις επόμενες **M** γραμμές της εισόδου έναν θετικό φυσικό αριθμό **X** και αναζητεί το **X** στην λίστα με τη λειτουργία `searchMF`.
- Τυπώνει έναν φυσικό αριθμό που αντιστοιχεί στο άθροισμα των τιμών που επιστρέφει η `searchMF` για όλες τις αναζητήσεις.

Παραδείγματα εισόδου:

```
3
1 1
1 2
1 3
2
1
2
```

```
3
1 3
2 2
3 1
2
1
3
```

```
3
1 3
2 2
2 1
2
2
2
```

Παραδείγματα εξόδου:

```
6
```

```
5
```

```
4
```

► Να υποβληθεί στο αυτόματο σύστημα υποβολής και ελέγχου μέχρι την Κυριακή 17/1/2021

Άσκηση 21.

Έστω ο **αφηρημένος τύπος δεδομένων** `bstree` για την παράσταση ενός δυαδικού δέντρου αναζήτησης (ΔΔΑ) ακεραίων αριθμών, ο οποίος υποστηρίζει τις ακόλουθες λειτουργίες:

```
class bstree {
public:
        bstree    ();        /* κατασκευαστής: κατασκευάζει ένα κενό ΔΔΑ  */
  int   height    ();        /* επιστρέφει το ύψος του ΔΔΑ (το κενό ΔΔΑ έχει ύψος 0)  */
  void  insert    (int x);   /* εισάγει τον αριθμό x στο ΔΔΑ  */
  int   search    (int x);   /* ψάχνει τον αριθμό x στο ΔΔΑ και επιστρέφει το επίπεδο στο οποίο
                                βρίσκεται (η ρίζα βρίσκεται στο επίπεδο 1) ή 0 αν δεν υπάρχει  */
  int   min       ();        /* επιστρέφει το ελάχιστο στοιχείο του ΔΔΑ  */
  int   max       ();        /* επιστρέφει το μέγιστο στοιχείο του ΔΔΑ  */
  void  inorder   ();        /* εκτυπώνει τα στοιχεία του ΔΔΑ με ενδοδιατεταγμένη διάσχιση  */
  void  preorder  ();        /* εκτυπώνει τα στοιχεία του ΔΔΑ με προδιατεταγμένη διάσχιση  */
  void  postorder ();        /* εκτυπώνει τα στοιχεία του ΔΔΑ με μεταδιατεταγμένη διάσχιση  */
};
```

Αφού υλοποίησετε αυτόν τον αφηρημένο τύπο δεδομένων, να χρησιμοποιήστε την υλοποίησή σας για να γράψετε ένα πρόγραμμα που εισάγει αριθμούς σε ένα ΔΔΑ και στη συνέχεια αναζητά πληροφορίες.

Το πρόγραμμα θα διαβάζει:

- από την πρώτη γραμμή της εισόδου το πλήθος **N** των ακεραίων που θα εισαχθούν στο ΔΔΑ,
- από τη δεύτερη γραμμή τους **N** ακεραίους προς εισαγωγή (διαφορετικούς μεταξύ τους),
- από την τρίτη γραμμή το πλήθος **M** των ακεραίων που θα αναζητηθούν στο ΔΔΑ, και
- από την τέταρτη γραμμή τους **M** ακέραιους προς αναζήτηση.

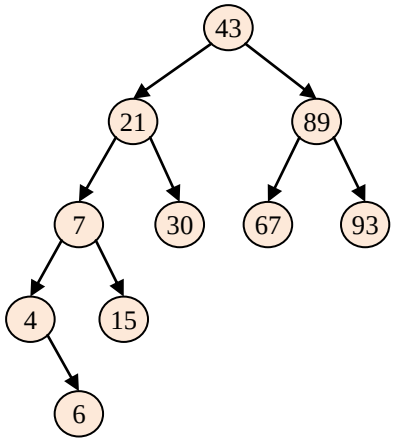
Το πρόγραμμα θα τυπώνει:

- στην πρώτη γραμμή το ύψος του ΔΔΑ που δημιουργήθηκε,
- στη δεύτερη γραμμή τον ελάχιστο και το μέγιστο ακέραιο του ΔΔΑ, χωρισμένους με ένα κενό διάστημα,

- στην τρίτη γραμμή **M** αριθμούς, χωρισμένους ανά δύο με ένα κενό διάστημα, που θα είναι με τη σειρά τα αποτελέσματα των αναζητήσεων (κλήσεων στη μέθοδο search),
- στην τέταρτη γραμμή τα στοιχεία του ΔΔΑ χωρισμένα με ένα κενό διάστημα, με ενδοδιατεταγμένη διάσχιση, ακολουθούμενα από τη λέξη "`end`".
- στην πέμπτη γραμμή τα στοιχεία του ΔΔΑ χωρισμένα με ένα κενό διάστημα, με προδιατεταγμένη διάσχιση, ακολουθούμενα από τη λέξη "`end`".
- στην έκτη γραμμή τα στοιχεία του ΔΔΑ χωρισμένα με ένα κενό διάστημα, με μεταδιατεταγμένη διάσχιση, ακολουθούμενα από τη λέξη "`end`".

Παράδειγμα εισόδου

```
10
43 21 7 30 15 89 67 4 93 6
3
15 43 42
```



Παράδειγμα εξόδου

```
5
4 93
4 1 0
4 6 7 15 21 30 43 67 89 93 end
43 21 7 4 6 15 30 89 67 93 end
6 4 15 7 30 21 67 93 89 43 end
```

► Να υποβληθεί στο αυτόματο σύστημα υποβολής και ελέγχου μέχρι την Κυριακή 17/1/2021