Θέμα 1.

Θεωρούμε το παρακάτω πρόγραμμα. Τί τυπώνεται κατά την εκτέλεσή του;

```
class sq {
protected:
    double n, s;
    void upd() {s=n*n;}
public:
    sq(): n(0), s(0) {}
    sq(double x): n(x) {upd();}
    void set(double d) {n=d; upd();}
    double N() {return n;}
    double S() {return s;}
};
class cb : public sq {
protected:
    double c;
    void upd() {s=n*n; c=s*n;}
public:
    cb(): c(0){}
    cb(double x): sq(x) {c=n*n*n;}
    double C() {return c;}
};
int main () {
    cb *myc = new cb(2);
    sq *mys = myc;
    cout<<mys->N()<<" "<< mys->S()<<" "<< myc->C()<< " ";
    myc->set(3);
    cout<<myc->N()<<" "<<myc->S()<<" "<<myc->C();
}
```

Α. 2 4 8 3 9 8 Β. 2 4 8 3 9 27 Γ. 2 4 8 3 9 0 Δ. 2 4 0 3 9 0

Θέμα 2.

Θεωρούμε το παρακάτω πρόγραμμα. Τί τυπώνεται κατά την εκτέλεσή του;

```
class A{
public:
    A() {cout << "A";}
    ~A() {cout<< "a";}
    void o() {cout << "@";}
};
int main () {
    B b;
    A *c = &b;
    c->o();
    b.o();
}
```

```
class B: public A {
public:
    B() {cout << "B";}
    ~B() {cout << "b";}
    void o() {cout<<"#";};
};
```

Α. ABA@#baa Β. ABA@#ba Γ. AB@#ba Δ. AaB@#ba

Θέμα 3.

Θεωρούμε το παρακάτω πρόγραμμα. Τί τυπώνεται κατά την εκτέλεσή του;

```
class Q {
protected:
    int dataq;
public:
    Q(): dataq(0) {}
    Q(int _d): dataq(_d) {throw "Q";}
    int get() {return dataq;}
};
class P: public Q {
protected:
    int datap;
public:
    P() {datap=1;}
    P(int _p): datap(_p) {throw datap;}
    int get() {return datap;}
};
```

```
int main() {
    try {
        Q qq;
        cout << qq.get();
        P ss;
        cout << ss.get();
        P pp(1);
        cout << pp.get();
        P rr(2);
        cout << rr.get();
    }
    catch (const char *a) {cout << a;
    catch (const int i) {cout << i;}
}
```

Α. 011 Β. 0112 Γ. 011Q Δ. 01Q

Θέμα 4.

Θεωρούμε το παρακάτω πρόγραμμα. Τί τυπώνεται κατά την εκτέλεσή του;

```
class S {
private:
    static int s, n;
    int data;
public:
    static void f(int a) { s += a; n++; }
    S(int b): data(b) { f(b);}
    S(): data(0) { f(data);}
    void status() { cout << s << " " << n << " "; }
};
int S::s = 0;
int S::n = 0;
```

```
int main() {
    T c1(1);
    c1.status();
    c1.f(1);
    T c2(2);
    c2.status();
    c2.f(3);
    T c3;
    c3.status();
}
```

Α. 1 1 4 3 6 4 Β. 1 1 4 3 6 5 Γ. 0 0 1 1 3 3 Δ. Κάτι άλλο

Θέμα 5.

Έχουμε ορίσει μία κλάση τα αντικείμενα της οποίας παριστάνουν ημερομηνίες και θέλουμε να υλοποιήσουμε τη σύγκριση ημερομηνιών ορίζοντας κατάλληλα τον `operator<`.

Α. Μπορούμε να τον ορίσουμε ως φίλη (friend) συνάρτηση με δύο παραμέτρους
Β. Μπορούμε να τον ορίσουμε ως μέθοδο της κλάσης μας με μία παράμετρο
Γ. Μπορούμε να τον ορίσουμε ως μέθοδο της κλάσης μας με δύο παραμέτρους
Δ. Μπορούμε να κάνουμε οποιοδήποτε από τα Α ή Β

Θέμα 6.

Θέλουμε μια αποδοτική δομή δεδομένων για να αποθηκεύσουμε ένα πολύ μεγάλο πλήθος μοναδικών strings τα οποία στη συνέχεια να αναζητούμε και να εμφανίζουμε ταξινομημένα. Είναι καλή ιδέα να χρησιμοποιήσουμε STL και συγκεκριμένα:

Α. map
Β. multimap
Γ. set
Δ. multiset

Θέμα 7.

Θέλουμε μια αποδοτική δομή δεδομένων για να αποθηκεύσουμε ένα πολύ μεγάλο πλήθος μη μοναδικών δεδομένων ενός τύπου Α, τα οποία να εμφανίζουμε με διαφορετικές, κατά περίπτωση, ταξινομήσεις. Μια αποδοτική πρακτική με χρήση STL είναι η ακόλουθη:

Α. vector<A> με κατάλληλες συναρτήσεις για τις διάφορες ταξινομήσεις
Β. multiset<A> με κατάλληλους, περισσότερους από έναν, υπερφορτωμένους τελεστές operator<
Γ. vector<A> με κατάλληλους, περισσότερους από έναν, υπερφορτωμένους τελεστές operator<
Δ. set<A> με κατάλληλες συναρτήσεις για τις διάφορες ταξινομήσεις

Θέμα 8.

Ποια/ποιες διασχίσεις εμφανίζουν τα στοιχεία ενός δυαδικού δένδρου αναζήτησης (BST) σε αύξουσα διάταξη;

Α. ενδοδιατεταγμένη (in-order)
Β. ενδοδιατεταγμένη (in-order) και προδιατεταγμένη (pre-order)
Γ. μεταδιατεταγμένη (post-order)
Δ. ενδοδιατεταγμένη (in-order) και μεταδιατεταγμένη (post-order)

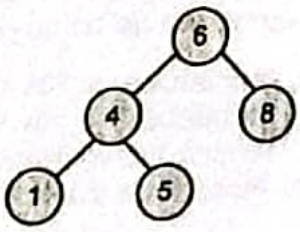
Θέμα 9.

Η διάσχιση κατά πλάτος (level-order) ενός δυαδικού δένδρου αναζήτησης είναι η ακόλουθη: 5 2 7 3 10. Ποια είναι η ενδοδιατεταγμένη διάσχιση του ίδιου δένδρου;

Α. 5 2 3 7 6 10
Β. 2 3 5 6 7 10
Γ. 3 2 6 10 7 5
Δ. Καμία από τις προηγούμενες

Θέμα 10.

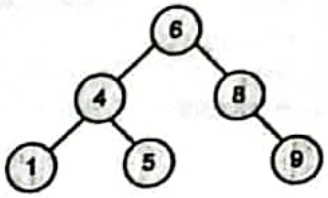
Στο δένδρο AVL του παρακάτω σχήματος διαγράφεται το κλειδί 8. Ποια πρόταση που αφορά τον κόμβο με το κλειδί 6 είναι ορθή (ελέγχουμε την ετικέτα του κόμβου που έχει το κλειδί 6 στο αρχικό AVL δένδρο και στο τελικό AVL δένδρο);



Α. Θα αλλάξει ετικέτα, θα γίνει αριστερά ψηλός.
Β. Θα αλλάξει ετικέτα, θα γίνει δεξιά ψηλός.
Γ. Θα αλλάξει ετικέτα, θα γίνει ίσα ψηλός.
Δ. Θα παραμείνει με την ίδια ετικέτα.

Θέμα 11.

Στο δένδρο AVL του διπλανού σχήματος διαγράφεται το κλειδί 6. Ποια πρόταση που αφορά τον κόμβο με το κλειδί 4 είναι ορθή (ελέγχουμε την ετικέτα του κόμβου που έχει το κλειδί 4 στο αρχικό AVL δένδρο και στο τελικό AVL δένδρο);



Α. Θα αλλάξει ετικέτα, θα γίνει αριστερά ψηλός.
Β. Θα αλλάξει ετικέτα, θα γίνει δεξιά ψηλός.
Γ. Θα αλλάξει ετικέτα, θα γίνει ίσα ψηλός.
Δ. Θα παραμείνει με την ίδια ετικέτα.

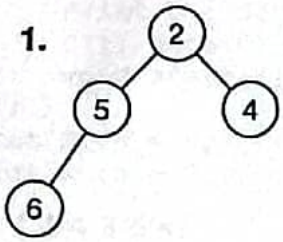
Θέμα 12.

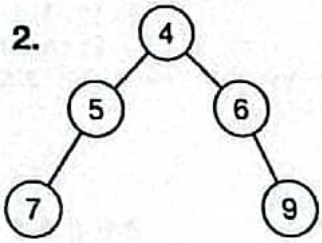
Δίνεται ένας πίνακας κατακερματισμού 15 θέσεων, για τον οποίο επιλέγεται η μέθοδος της διαίρεσης (mod15) ως μέθοδος κατακερματισμού με αλυσίδωση. Έστω ότι επιθυμούμε να αποθηκεύσουμε κλειδιά που μπορούν να πάρουν τιμές στο διάστημα των ακεραίων [0,50]. Ποια από τις παρακάτω ακολουθίες εισαγωγής κλειδιών θα οδηγήσει σε σχηματισμό αλυσίδας μήκους τουλάχιστον 3 σε κάποια θέση του πίνακα;

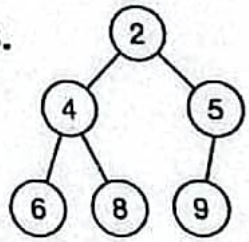
Α. 3, 4, 5, 6, ..., 30
Β. 5, 6, 7, 8, ..., 24
Γ. 1, 4, 7, 10, ..., 43
Δ. καμμία από τις παραπάνω.

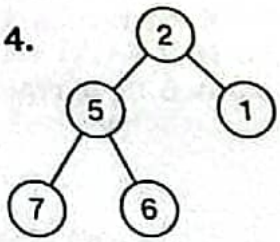
Θέμα 13.

Δίνονται τα παρακάτω δένδρα που απεικονίζουν σύνολα σε μια δομή union-find. Ποια από αυτά τα δένδρα μπορεί να προκύψουν με διαδοχικές εφαρμογές της διαδικασίας union by size; (θεωρήστε ότι ο αλγόριθμος ξεκινάει με τα μονοσύνολα: {1}, {2}, {4}, {5}, {6}, {7}, {8}, {9})

1.


2.


3.


4.


Α. Τα 1 και 3. Β. Τα 3 και 4. Γ. Μόνο το 4. Δ. Μόνο το 1.

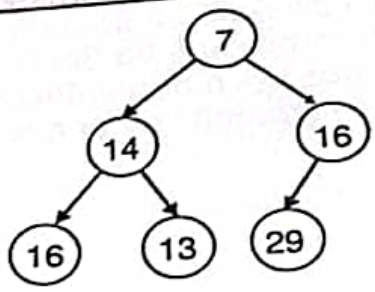
Θέμα 14.
Είχε να κάνει με εισαγωγή/διαγραφή από σωρό.

Θέμα 15.
Δίνεται ένα Β δέντρο 3 οδεύσεων που έχει έναν κόμβο ρίζα με 2 αριθμούς και 3 κόμβους παιδιά με 2 αριθμούς το καθένα.
Υπάρχει αριθμός που μπορείς να προσθέσεις στο δέντρο χωρίς να αυξηθεί το ύψος του;

Μέρος Β. Ελεύθερα ερωτήματα

Απαντήστε στο κενό κάτω από το ερώτημα.

Θέμα 16. Σωροί (5 μονάδες)

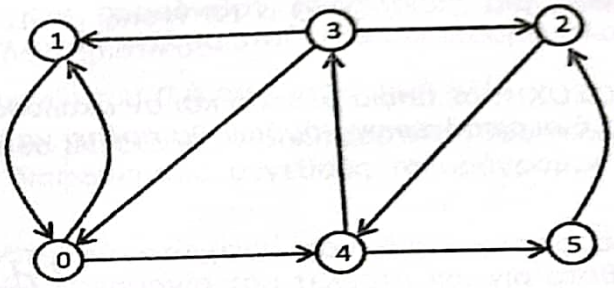


νεται ο σωρός ελαχίστου του διπλανού σχήματος:
Δώστε τη μορφή του σωρού μετά την εισαγωγή στοιχείου με τιμή **12.**
ες συγκρίσεις θα πραγματοποιηθούν;

στω ότι τώρα διαγράφουμε το ελάχιστο στοιχείο (**αφού προηγηθεί σαγωγή του στοιχείου 12**). Δώστε τη μορφή του σωρού μετά από ιαγραφή αυτή. Ποιες συγκρίσεις θα πραγματοποιηθούν;

Θέμα 17. Αλγόριθμοι BFS/DFS (5 μονάδες)

Δίνεται ο γράφος του διπλανού σχήματος.



α) Εκτελέστε τον αλγόριθμο εξερεύνησης κατά πλάτος (BFS) στον γράφο αυτό με αφετηρία τον κόμβο **4**.

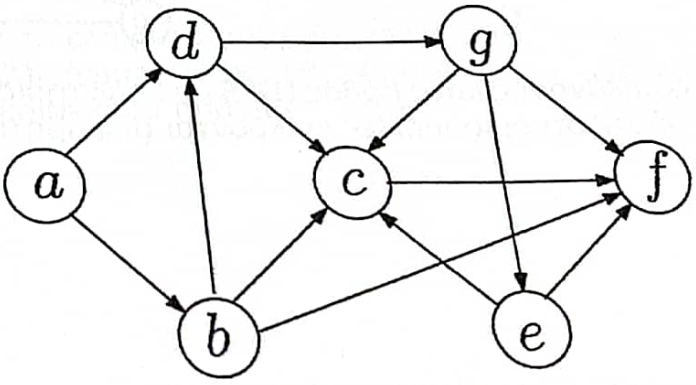
Δείξτε ***μόνο*** την σειρά επίσκεψης των κόμβων (θεωρήστε ότι οι ισοπαλίες επιλύονται αλφαβητικά) και τα περιεχόμενα της ουράς.

β) Εκτελέστε τον αλγόριθμο εξερεύνησης κατά βάθος (DFS) με αφετηρία τον κόμβο **0**. Δείξτε την σειρά επίσκεψης των κόμβων (θεωρήστε ότι οι ισοπαλίες επιλύονται αλφαβητικά). Ποιες back edges θα βρει ο αλγόριθμος;

Θέμα 18. Τοπολογική ταξινόμηση (5 μονάδες)

Δώστε μία τοπολογική ταξινόμηση για τον γράφο του σχήματος. Αναφέρετε τη μέθοδο που ακολουθήσατε και εξηγήστε με λίγα λόγια γιατί δίνει το αποτέλεσμα που βρήκατε.

ΠΡΟΣΟΧΗ: *σε όποια μέθοδο και αν ακολουθήσετε, οποτεδήποτε υπάρχει δυνατότητα επιλογής μεταξύ δύο ή περισσότερων κόμβων θα πρέπει να επιλέξετε αυτόν που προηγείται αλφαβητικά.*



Θέμα 19. Διανύσματα (10 μονάδες)

Υλοποιήστε την κλάση my_vector χωρίς τη χρήση της STL η οποία θα αναπαριστά μονοδιάστατα διανύσματα ακεραίων μεγέθους size και θα έχει τα εξής χαρακτηριστικά:

α) Ένα διάνυσμα θα μπορεί να αρχικοποιηθεί με όλα τα στοιχεία του *0* ή σε κάποια τιμή *val*.

β) Θα πρέπει να υποστηρίζεται η πρόσθεση διανυσμάτων ίδιου μεγέθους με υπερφόρτωση του τελεστή "+". Σε περίπτωση που ζητηθεί πρόσθεση διανυσμάτων διαφορετικού μεγέθους το πρόγραμμα θα πρέπει να «ρίχνει» εξαίρεση.

γ) Θα πρέπει να υποστηρίζεται η προσπέλαση του "*k*"-οστού στοιχείου του διανύσματος μέσω υπερφόρτωσης του τελεστή [] και να υποστηρίζεται η ορθή λειτουργία του τελεστή και για σταθερά διανύσματα. Σε περίπτωση που *k >= size* το πρόγραμμα θα πρέπει να «ρίχνει» εξαίρεση.

δ) Θα πρέπει να υποστηρίζεται εκτύπωση του διανύσματος με υπερφόρτωση του τελεστή <<.

ε) Αν κρίνετε ότι χρειάζεται να υλοποιηθούν καταστροφέας, κατασκευαστής αντιγραφής και τελεστής ανάθεσης, να αιτιολογήσετε την ανάγκη και να δώσετε τις σχετικές υλοποιήσεις. Διαφορετικά αιτιολογήστε την μη ανάγκη υλοποίησης.

Θέμα 20. Αναζήτηση ονομάτων (10 μονάδες)

Από την είσοδο διαβάζεται ένα τυχαίο κείμενο. Στο αρχείο words.txt περιέχονται μη-επαναλαμβανόμενες επιλεγμένες λέξεις. Να γράψετε ένα κομψό και αποδοτικό πρόγραμμα που μετράει τη συχνότητα εμφάνισης στην είσοδο των λέξεων που περιέχονται στο αρχείο words.txt, και την εμφανίζει στην έξοδο, ακολουθούμενη από το πλήθος των λέξεων της εισόδου που δεν περιέχονται στο αρχείο words.txt, όπως φαίνεται στο παράδειγμα. Κάθε ίδια λέξη που δεν περιέχεται στο words.txt προσμετράται μόνο μία φορά, ανεξάρτητα από το πλήθος των εμφανίσεών της στην είσοδο.

```
$ cat words.txt
alpha bravo charlie delta echo
november uniform sierra mike

$ cat input.txt
november tango uniform alpha echo
charlie echo india sierra tango hotel
echo bravo echo sierra tango tango
hotel echo sierra uniform mike mike
echo romeo india sierra charlie lima
oscar sierra echo
```

```
$ ./a.out < input.txt
alpha 1
bravo 1
charlie 2
echo 7
mike 2
november 1
sierra 5
uniform 2
6
```

Υπόδειξη: Θα χρειαστείτε κατάλληλη κλάση της βιβλιοθήκης STL για την αποθήκευση των λέξεων του αρχείου words.txt με τρόπο που να είναι αποδοτική η αναζήτησή τους. Επίσης θα χρειαστείτε μια απεικόνιση κάθε λέξης της εισόδου στο πλήθος των εμφανίσεών της. Με παρόμοιο τρόπο μπορείτε να μετρήσετε και τις λέξεις της εισόδου που δεν περιέχονται στο words.txt.

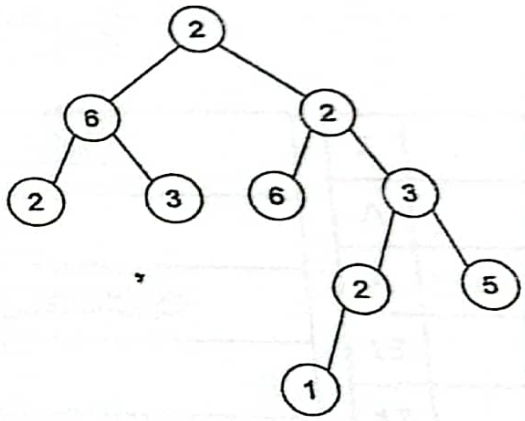
Βαθμολογία: συχνότητα λέξεων: 6/10, πλήθος άγνωστων λέξεων 4/10

Θέμα 21. Δυαδικό δένδρο (10 μονάδες)

Ορίστε τον τύπο του κόμβου **node** του δυαδικού δένδρου που περιέχει ως πληροφορία τιμές οποιουδήποτε τύπου **T**.

Στη συνέχεια, γράψτε μία κομψή και αποδοτική συνάρτηση (template) η οποία να δέχεται ως παράμετρο ένα τέτοιο δένδρο **t**, μία τιμή **x** του τύπου **T** και έναν ακέραιο **h**. Η συνάρτησή σας να επιστρέφει το πλήθος των στοιχείων που έχουν τιμή ίση με x και βρίσκονται σε ύψος h. Μπορείτε να θεωρήσετε ότι το ύψος της ρίζας είναι είτε 0 είτε 1.

Τι πρέπει να έχει οριστεί για να λειτουργεί για κάθε τύπο **T** η συνάρτησή σας;



Παράδειγμα: Για το δέντρο του διπλανού σχήματος (όπου T=int) και για x=3, h=3 η συνάρτησή σας πρέπει να επιστρέφει 2 (θεωρώντας ότι η ρίζα βρίσκεται σε ύψος 1).

Βαθμολογία: σωστός ορισμός templates: 4/10, υπολογισμός: 6/10