ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10:ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 1 Φεβρουαρίου 2013

1. (α′) Τι επιτυγχάνουμε με το μηχανισμό του εγκλωβισμού (encapsulation) στον αντικειμενοστραφή προγραμματισμό, δεδομένου ότι αυτός επιτελεί απόκρυψη πληροφορίας; Που αυτός είναι χρήσιμος στον προγραμματισμό, γενικώτερα;

   (β′) Δίδεται το παρακάτω πρόγραμμα C++

```
#include<iostream>
using namespace std;

class A {
    protected: int data;
    public:
      A(int i = 0) : data(i)
        { cout << "I am constructing an A with " << i << endl; }
      ~A() { cout << "Destructing an A with " << data << endl; }
      virtual void bla() { cout << "Calling A::bla" << endl; data++; }
      void bloo() { cout << "Calling A::bloo" << endl; }
      void fun() { cout << "Calling A::fun" << endl; data++; } };

class B : public A { int data;
    public:
      B(int i = 100) : data(i)
        { cout << "I am constructing a B with " << i << endl; }
      ~B() { cout << "Destructing an B with" << data << endl; }
      void bla() { cout << "Calling B::bla" << endl; data++;}
      void bloo() { cout << "Calling B::bloo " << endl; A::data++; } };

// void foo(A param){
// void foo(A& param){
// void foo(B param){
// void foo(B& param){
     param.bla(); param.bloo(); param.fun(); }

int main(){
//  A data; foo(data);
//  B data; foo(data);
    return 0;  }
```

   Αποσχολιάζοντας μια από τις δύο σχολιασμένες γραμμές στο σώμα της `main` τη φορά, ποια/ες γραμμές μπορούν να αποσχολιαστούν από τη συνάρτηση `foo` για να μπορεί να εκτελεστεί το πρόγραμμα; Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης.

2. Δίδεται το παρακάτω πρόγραμμα C++:

```
#include<iostream>
using namespace std;

class A1 { int data;
   public:
     A1(int i = 10 ) : data(i)
```

```
        { cout << "I am constructing an A1 with: " << i << endl; }
     A1(const A1& a1) : data(a1.data)
        { cout << "I am copy constructing an A1" << endl; }
     ~A1() { cout << "I am destroying an A1 with: " << data << endl; }
     void change() { data = data * 10; } };

class A2 { int data;
   public:
     A2(int i = 20 ) : data(i)
        { cout << "I am constructing an A2 with: " << i << endl; }
     A2(const A2& a2) : data(a2.data)
        { cout << "I am copy constructing an A2" << endl; }
     ~A2() { cout << "I am destroying an A2 with: " << data << endl; }
     void change() { data = data * 20; } };

class A3 {
   public:
     A3() { cout << "I am constructing an A3" << endl; }
     A3(const A3& a3)
        { cout << "I am copy constructing an A3" << endl; }
     ~A3() { cout << "I am destroying an A3" << endl; }
     void change() { cout << "Nothing to change" << endl; } };

class A { A1 a1; A2 a2; A3 a3;
   public:
     A() { cout << "I am constructing an A" << endl; }
     A(const A& a) : a1(a.a1)
        { cout << "I am copy constructing an A" << endl; }
     ~A() { cout << "I am destroying an A" << endl; }
     A& operator=(const A& a)
        { cout << "I am performing a stupid assignment between As" << endl;
          if (this != &a) a1 = a.a1;
          return *this ;}
     void change() { a1.change(); a2.change(); a3.change(); } };

class BigA { A data1; A& data2;
     public:
      BigA(A& a): data1(a), data2(a)
          { cout << "I just constructed a BigA" << endl; }
     ~BigA() { cout << "I am destroying a BigA" << endl; }
      A get(int index) { if (index == 1) return data1;
                         else return data2; } };

// BigA volta(BigA& biga)
// BigA& volta(BigA& biga)
           { cout << "Volta ta data?" << endl;
             return biga; }

int main() {
    A first; A second(first); A third = first;
    third.change(); second = third;
```

```
BigA biga(first);
volta(biga).get(2).change();

return 0; }
```

Αποσχολιάζοντας μια από τις δύο σχολιασμένες γραμμές τη φορά, δώστε το αποτέλεσμα της εκτέλεσης, αιτιολογώντας την απάντησή σας.

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία προσομοίωση παιδικής χαράς. Η παιδική χαρά έχει μια μέγιστη χωρητικότητα παιδιών —θεωρούμε μόνο παιδιά—. Αποτελείται από πέντε τραμπάλες. Επίσης, τα παιδιά μπορούν να παίξουν διελκυστίνδα.

   Στις τραμπάλες, υποχρεωτικά, κάθεται αγόρι με αγόρι και κορίτσι με κορίτσι.

   Στη διελκυστίνδα, τα παιδιά χωρίζονται σε δύο ομάδες και φτιάχνουν δύο αλυσίδες, τραβώντας το ένα το άλλο. Σε κάθε αλυσίδα, το καθένα παιδί τραβιέται από ένα άλλο εκτός από το τελευταίο που δεν τραβιέται από κανένα. Επίσης, το πρώτο τραβά ένα σκοινί από μια άκρη. Την άλλη άκρη του σκοινιού τραβά το πρώτο παιδί της άλλης αλυσίδας. Ο νικητής είναι η ομάδα που, με το τράβηγμα, θα αποσπάσει το σκοινί. Στη διελκυστίνδα, τα παιδιά διατάσσονται στην κάθε αλυσίδα κατά φθίνουσα σειρά δύναμης —το πιο δυνατό τραβά το σκοινί—. Σε κάθε χρονική στιγμή, η μια αλυσίδα μπορεί να έχει το πολύ ένα παιδί περισσότερο από την άλλη. Επίσης, υπάρχει ένα μέγιστο πλήθος παιδιών που μπορούν να παίξουν.

   Για την προσομοίωση να χρησιμοποιηθούν οι κλάσεις "Παιδική Χαρά" (`Playground`), "Διελκυστίνδα" (`War`), "Τραμπάλες" (`Seesaws`) και "Τραμπάλα" (`Seesaw`).

   Η κλάση "Παιδική Χαρά":

   - έχει μια μέγιστη χωρητικότητα παιδιών (`capacity`)
   - έχει ένα πλήθος παιδιών που είναι μέσα (`children_in`)
   - έχει 5 τραμπάλες (`seesaws`)
   - έχει μια ένδειξη σε διελκυστίνδα (`war`)

   Η κλάση "Παιδική Χαρά" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά, καταχωρείται η χωρητικότητά της, είναι άδεια από παιδιά, οπότε δεν έχει σχηματιστεί και διελκυστίνδα.
   - Ένα παιδί μπαίνει στην παιδική χαρά, αν χωράει. Αν δεν χωράει, εκτυπώνεται σχετικό μήνυμα. Αν χώρεσε, αυξάνεται το πλήθος παιδιών που είναι μέσα κατά μια μονάδα. Αρχικά δοκιμάζει να μπει στη διελκυστίνδα. Αν δε χωράει, δοκιμάζει να μπει στις τραμπάλες (`enter`)
   - Γίνεται λήξη διελκυστίνδας, αναδεικνύοντας ποια είναι η νικήτρια αλυσίδα και μηδενίζοντας την ένδειξη στη διελκυστίνδα (`war_end`)

   Η κλάση "Διελκυστίνδα":

   - έχει μια μέγιστη χωρητικότητα παιδιών (`capacity`)
   - έχει δύο σύνολα παιδιών (`left_children`, `right_children`)

   Η κλάση "Διελκυστίνδα" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά καταχωρείται η χωρητικότητά της και τα σύνολα παιδιών είναι κενά.
   - Ένα παιδί μπαίνει στη διελκυστίνδα, αν δεν υπερβαίνει τη χωρητικότητά της, τοποθετούμενο στην κατάλληλη θέση με βάση τη δύναμή του σε μια από τις δύο αλυσίδες (`enter`)
   - Αναδεικνύεται η νικήτρια αλυσίδα, όπως προκύπτει από το άθροισμα των δυνάμεων των παιδιών που την αποτελούν (`decide_winner`)

Η κλάση "Τραμπάλες":

- έχει 5 τραμπάλες (`seesaws`)

Η κλάση "Τραμπάλες" χαρακτηρίζεται από την εξής συμπεριφορά:

- Ένα παιδί μπαίνει στις τραμπάλες, αν μπορεί να μπει σε μια τραμπάλα. Αν δεν υπάρχει διαθέσιμη τραμπάλα, εκτυπώνεται σχετικό μήνυμα (`enter`)

Η κλάση "Τραμπάλα":

- έχει δύο ενδείξεις σε παιδιά (`left_child`, `right_child`)

Η κλάση "Τραμπάλα" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά, είναι κενή από παιδιά.
- Ένα παιδί μπαίνει στην τραμπάλα, αν του είναι δυνατό —είτε είναι άδεια είτε μόνο η μια θέση είναι κατειλημμένη και το παιδί που την καταλαμβάνει είναι του ίδιου φύλου—. Αν η εισαγωγή είναι επιτυχής, καταχωρείται στην τραμπάλα μια ένδειξη προς αυτό το παιδί (`enter`)

Τα "Παιδιά" δομούνται από το φύλο τους (`M` ή `F`) και τη δύναμή τους (ακέραιος).

Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις. Θεωρήστε δεδομένο έναν τύπο "λίστας " όπως σας βολεύει, απλώς τεκμηριώστε τον.
Όπου θεωρείτε απαραίτητο, κάνετε παραδοχές στις προδιαγραφές, τεκμηριώνοντάς τις.

Σημείωση: Στα θέματα στα οποία σας ζητείται να βρείτε αποτελέσματα, όποτε αυτά επαναλαμβάνονται ή η αιτιολόγηση είναι ίδια με κάτι που ήδη έχει αιτιολογηθεί, μην επαναλαμβάνεστε. Απλά κάντε τη σχετική αναφορά ομοιότητας.