ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10:ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 20 Σεπτεμβρίου 2018

1. (α′) Στον αντικειμενοστραφή προγραμματισμό μιλάμε για "αφαίρεση στα δεδομένα" καθώς και για "σύνθεση" και "κληρονομικότητα" μεταξύ κλάσεων. Πού αναφέρεται ο καθένας όρος από αυτούς; Δώστε ένα παράδειγμα όπου η αφαίρεση στα δεδομένα είναι χρήσιμη στην υλοποίησή του αλλά η κληρονομικότητα δεν είναι απαραίτητη.

   (β′) Δίδεται το παρακάτω πρόγραμμα C++:

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

class A{ int data;
  public:
    A(int i=100) : data(i)
       { cout << "I just constructed an A with: " << data << endl; }
    A(const A& a) : data(a.data)
       { cout << "I just constructed an A by copying with data "
              << data << endl; }
    ~A() { cout << "Destructing an A with data " << data << endl; }
     int get_data() { return data; }
     void change() { data += 100; } };

class FatA{
     A data1; A* data2; A& data3;
   public:
     FatA(A& a) : data1(a), data3(a)
       {  data2 = new A(a);
          cout << "I just constructed a FatA with: "
               << data1.get_data() << "  " << data2->get_data() << "  "
               << data3.get_data() << endl; }
     FatA(FatA& fa) : data1(fa.data1), data3(fa.data3)
          { data2 = new A(*(fa.data2)); }
     ~FatA() { cout << "Destructing a FatA " << endl; delete data2; }
      void change() { data1.change(); data2->change(); data3.change();} };


int main(){ A a;
            FatA f1(a); FatA f2 = f1;
            f1.change(); f2.change();
            return 0; }
```

   και το παρακάτω πρόγραμμα Java:

```
public class first {
   public static void main(String[] args) {
        A a = new A(100);
        FatA f1 = new FatA(a); FatA f2 = f1;
        f1.change(); f2.change();
        System.out.println("Printing f1 data"); f1.print();
        System.out.println("Printing f2 data"); f2.print(); } }
```

```
class A{
private int data;
A(int i){ data = i;
          System.out.println("I just constructed an A with " + data); }
int get_data(){ return data;}
void change(){ data += 100; } }

class FatA{
   private A data1;
   private A data2;
   private A data3;

   FatA(A a)
     {  data1 = a; data2 = a; data3 = a;
        System.out.println("I just constructed a FatA with: " +
             data1.get_data() + "  " + data2.get_data() + "  " +
             data3.get_data()); }
    void change() { data1.change(); data2.change(); data3.change(); }
    void print() {System.out.println( " The FatA data are: " +
                                       data1.get_data() + "  " +
                                       data2.get_data() + " " +
                                       data3.get_data());} }
```

Δώστε τα αποτελέσματα της εκτέλεσης των παραπάνω προγραμμάτων, αιτιολογώντας τα. Εντοπίστε τις ουσιαστικές διαφορές στα αποτελέσματα αυτά και αιτιολογείστε τις.

2. Δίδεται το παρακάτω πρόγραμμα C++. Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσής του, αποσχολιάζοντας μιά γραμμή κάθε φορά στον ορισμό της κλάσης `Musical`.

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

class Actor{ int strength;
  public :
   Actor(int s=10) : strength(s) {cout << "Creating an Actor" << endl;}
   Actor(const Actor& actor) : strength(actor.strength)
         {cout << "Creating a Copy of an Actor" << endl;}
   virtual ~Actor() {cout << "Disappearing ..." << strength << endl;}
   int get_strength() {return strength;}
   void tiring() {if (strength > 0) strength-- ;}
   virtual void act() {cout << "words, words, words ..." << endl; tiring();} };
```

```
class ActorDancer : public Actor{ int strength;
  public:
   ActorDancer(int as=5, int s=100) : Actor(as), strength(s)
        {cout << "Creating an Actor who Dances" << endl;}
   virtual ~ActorDancer()
        {if (get_strength()> 80) cout << "Exiting but no problem at all" << endl ;
             else { if (get_strength()> 50) cout << "Exiting but still keeping" << endl ;
                        else cout << "Exiting exhausted" << endl;}
          cout << strength << endl;}
   int get_strength() {return Actor::get_strength() + strength;}
   void tiring() {cout << strength << endl;
                  if (strength > 20 ) strength -= 20;
                  cout << strength << endl;}
   virtual void act() {cout << "I am dancing in the streets" << endl; tiring();} };

class ActorSingerDancer : public ActorDancer{
  public:
    ActorSingerDancer(int astrength=30, int dstrength=50) :
                      ActorDancer(astrength, dstrength)
         {cout << "Creating an Actor who Sings and Dances" << endl;}
    virtual void act() {ActorDancer::act();
                        cout << "I am singing under the stars!" << endl;
                        tiring(); Actor::tiring();} };

class Musical{
   ActorSingerDancer& starring1; ActorSingerDancer& starring2;
   ActorDancer& dancer; Actor& actor;
  public :
    Musical(ActorSingerDancer& s1, ActorSingerDancer& s2, ActorDancer& d, Actor& a) :
            starring1(s1), starring2(s2), dancer(d), actor(a)
       {cout << "The show starts!" << endl;
        scene(s1,s2); scene(d,a);}
    ~Musical() {cout << "The End" << endl;}
//    void scene(Actor a1, Actor a2)
//    void scene(Actor& a1, Actor& a2)
               { a1.act(); a2.act();} };

int main(){ ActorSingerDancer ES(60); ActorSingerDancer RG(40,70);
            ActorDancer AD; Actor A;
            Musical la_la_land(ES, RG, AD, A);
    return 0; }
```

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία προσομοίωση ενός θερινού σινεμά. Έστω ότι η προσομοίωση αυτή θεωρεί τις εξής κλάσεις:

   - κλάση: "Ταμείο" που προσομοιώνει την εξυπηρέτηση στο ταμείο του θερινού
   - κλάση: "Bar" που προσομοιώνει την εξυπηρέτηση στο μπαρ του θερινού
   - κλάση: "Cinema" που προσομοιώνει την εξυπηρέτηση στο θερινό σινεμά.

   Η κλάση "Ταμείο":

   - έχει μία ουρά θεατών (`customers`)

   Η κλάση "Ταμείο" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά το ταμείο είναι άδειο.
   - Στο ταμείο, ένας θεατής προστίθεται (`add`) στο τέλος της ουράς του.
   - Στο ταμείο εξυπηρετείται (`serve`) ο πρώτος θεατής κάθε φορά, διαγράφοντάς τον από την ουρά.

   Η κλάση "Bar":

   - έχει μία ουρά θεατών (`customers`)
   - αποθηκεύει το τζίρο των πωλήσεων (`M`)

   Η κλάση "Bar" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά το μπαρ είναι άδειο.
   - Στο μπαρ, ένας θεατής προστίθεται (`add`) στο τέλος της ουράς του.
   - Στο μπαρ εξυπηρετείται (`serve`) ο πρώτος θεατής κάθε φορά, συμβάλλοντας στο τζίρο με το ποσό αγοράς του, και διαγράφεται από την ουρά.

   Η κλάση "Cinema":

   - έχει ένα ταμείο (`cashier`)
   - έχει ένα μπαρ (`bar`)
   - έχει μια σταθερά που αντιστοιχεί στο αντίτιμο του εισιτηρίου (`ticket_price`)
   - αποθηκεύει τον τρέχοντα συνολικό τζίρο του (`M`)

   Η κλάση "Cinema" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά, ανατίθεται το αντίτιμο του εισιτηρίου του και ο συνολικός τζίρος είναι μηδέν.
   - Κάνει εξυπηρέτηση, δημιουργώντας θεατή και προσθέτοντάς τον στο ταμείο. Κατόπιν, εξυπηρετεί το ταμείο, αυξάνοντας το συνολικό τζίρο κατά το αντίτιμο του εισιτηρίου. Αν ο θεατής που εξυπηρετήθηκε θέλει να αγοράσει κάτι από το μπαρ, τον προσθέτει στο μπαρ και τυχαία επιλέγει να εξυπηρετήσει το μπαρ ή όχι (`serve`)

   Ο θεατής χαρακτηρίζεται από το ποσό αγοράς του από το μπαρ. Το ποσό αυτό δημιουργείται τυχαία κατά τη δημιουργία του θεατή. Αν δεν θέλει να αγοράσει κάτι από το μπαρ, το ποσό αυτό είναι μηδέν.

   Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις, κάνοντας παραδοχές στις προδιαγραφές, όπου (και αν) το θεωρείτε απαραίτητο, τεκμηριώνοντάς τις. *Σημείωση:* Θεωρείστε δεδομένον ένα τύπο ουράς αλλά δώστε τις προδιαγραφές του.

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