ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10:ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 1 Αυγούστου 2014

1. (α′) Τι εννοούμε με τον όρο "αρχικοποίηση" αντικειμένων; Με ποιές συναρτήσεις-μέλη κάνουμε αρχικοποίηση αντικειμένων στη C++; Στους κώδικες που υπάρχουν στα θέματα, σημειώσετε, σε πλαίσια, ένα σημείο που γίνεται αρχικοποίηση αντικειμένου και τη συνάρτηση που καλείται.

   (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;
   public:
     FatA(A& a) : data1(a),data2(a)
       { cout << "I just constructed a FatA with: "
              << data1.get_data() << "  " << data2.get_data() << endl;}
     ~FatA() { cout << "Destructing a FatA " << endl;}
      void change() { data1.change(); data2.change(); } };


class BigFatA{
      FatA& data1; FatA& data2;
   public:
     BigFatA(FatA& fatA) : data1(fatA),data2(fatA)
       { cout << "I just constructed a BigFatA " << endl; }
     ~BigFatA() { cout << "Destructing a BigFatA " << endl;}
      void change() { FatA& rdata1 = data1; rdata1.change();
                      FatA cdata2 = data2; cdata2.change(); } };

int main(){
    A a;
    FatA fatA(a); fatA.change();

    BigFatA bigFatA(fatA);
    bigFatA.change();

   return 0; }
```

   Δώστε το αποτέλεσμα της εκτέλεσής του, αιτιολογώντας την απάντησή σας.

2. Δίδεται το παρακάτω πρόγραμμα C++

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

class Dish{
   protected:
      int cost;
      int cals;
   public:
      Dish(int cst=0, int cl=150) : cost(cst), cals(cl)
          { cout << "A Dish was created " << endl; }
      ~Dish() { cout << "A Dish will be destroyed" << endl; }
      virtual void eat() { cout << "Eating a Dish with cals " << cals
                         << " and cost " << cost << endl; }
      int get_cost() { return cost; }
      int get_cals() { return cals; } };

class FirstCourse : public Dish{
     public:
       FirstCourse(int cst, int cl) : Dish(cst,cl)
           {cout << "A FirstCourse was created with " <<
                     cost << " and " << cals << endl; }
      ~FirstCourse() { cout << "A FirstCourse will be destroyed" << endl; } };

class Starter : public FirstCourse{
      public :
      Starter(int cst, int cl) : FirstCourse(cst, cl)
           { cout << "A Starter was created " << endl ;}
      ~Starter() { cout << "A Starter will be destroyed" << endl; }
       void eat() { FirstCourse::eat();
                    cout << "Eating Starter " << endl; } };

class Salad : public FirstCourse{
      public :
      Salad(int cst, int cl) : FirstCourse(cst, cl)
           { cout << "A Salad was created " << endl ;}
      ~Salad() { cout << "A Salad will be destroyed" << endl; }
       void eat() { FirstCourse::eat();
                    cout << "Eating Salad " << endl; } };


class MainCourse : public Dish{
     public:
       MainCourse(int cst, int cl) : Dish(cst,cl)
           { cout << "A MainCourse was created with " <<
                     cost << " and " << cals << endl; }
      ~MainCourse() { cout << "A MainCourse will be destroyed" << endl; } };
```

```
class Dessert : public Dish{
     public:
       Dessert(int cst, int cl) : Dish(cst,cl)
           { cout << "A Dessert was created " << endl; }
      ~Dessert() { cout << "A Dessert will be destroyed" << endl; }
       void eat() { Dish::eat();
                    cout << "Eating Dessert " << endl; } };


class Meal{
   static const int service = 1;
   Dish starter;
   Dish salad;
   Dish maincourse;
   Dish dessert;

  public:
    Meal() { cout << "A Meal has been created " << endl; }
    ~Meal() { cout << "A Meal will be destroyed" << endl; }
    void order() { starter = *(new Starter(4, 100));
                   salad = *(new Salad(6, 50));
                   maincourse = *(new MainCourse(10, 500));
                   dessert = *(new Dessert(4, 450)); }
    void eat() { starter.eat(); salad.eat(); maincourse.eat(); dessert.eat();}
    int pay_bill() { return starter.get_cost() + salad.get_cost() +
                            maincourse.get_cost() + dessert.get_cost() + service;}
    int evaluate_cals() { return starter.get_cals() + salad.get_cals() +
                                 maincourse.get_cals() + dessert.get_cals(); } };


int main(){

   Meal meal;

   meal.order();
   meal.eat();

   cout << "You are going to pay " << meal.pay_bill() << " euros " << endl;
   cout << "You are going to get " << meal.evaluate_cals()
        << " calories " << endl ;
   return 0; }
```

Δώστε το αποτέλεσμα της εκτέλεσής του, αιτιολογώντας την απάντησή σας.

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία προσομοίωση της εξυπηρέτησης πελατών στα εκδοτήρια/ταμεία ενός Complex Cinema. Έστω ότι η προσομοίωση αυτή θεωρεί τις εξής κλάσεις:

   - κλάση: "Ταμείο" που προσομοιώνει την εξυπηρέτηση σ΄ ένα ταμείο/εκδοτήριο
   - κλάση: "Cinema" που προσομοιώνει την κίνηση των ταμείων του Complex Cinema

   Η κλάση "Ταμείο":

   - έχει μία ουρά πελατών, `customers`, όπου ο κάθε πελάτης αντιπροσωπεύεται μόνο από την ταινία που επιθυμεί να παρακολουθήσει (*Σημείωση:* για λόγους απλότητας, η τανία μπορεί να είναι ένας αριθμός από το 1 έως το 8)

   Η κλάση "Ταμείο" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά ένα ταμείο είναι άδειο.
   - Σε ένα ταμείο, ένας πελάτης προστίθεται (`add`) στο τέλος της ουράς του.
   - Στο ταμείο εξυπηρετείται (`serve`) ο πρώτος πελάτης κάθε φορά, συμβάλλοντας στο συνολικό τζίρο του Complex Cinema με το αντίτιμο του εισιτηρίου, και διαγράφεται από την ουρά.

   Η κλάση "Cinema":

   - έχει μια σταθερά που αντιστοιχεί στο αντίτιμο του εισιτηρίου
   - έχει ένα σύνολο από 10 ταμεία (`cashiers`)
   - αποθηκεύει τον τρέχοντα συνολικό τζίρο του, όπως έχει διαμορφωθεί από τα εισιτήρια που έχουν κοπεί απ΄ όλα τα ταμεία (`M`)

   Η κλάση "Cinema" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά, ανατίθεται το αντίτιμο του εισιτηρίου του
   - Αυξάνει το συνολικό τζίρο κατά το αντίτιμο του εισιτηρίου, όταν ένας πελάτης εξυπηρετείται από ένα ταμείο (`increase_amount`).
   - Αναγγέλλεται μια ταινία, δίδοντας τον αριθμό που την αναπαριστά, και τότε, για κάθε ταμείο, οι πελάτες που περιμένουν στην ουρά του και ενδιαφέρονται να δουν την ταινία αυτή, μετακινούνται στην αρχή της ουράς, διατηρώντας τη σχετική διάταξη μεταξύ τους (`announce_film`)

   Να υλοποιήσετε σε C++ τις δύο αυτές κλάσεις, κάνοντας παραδοχές στις προδιαγραφές, όπου (και αν) το θεωρείτε απαραίτητο, 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.