ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 14 Φεβρουαρίου 2008

1. (α′) Σχολιάστε επιγραμματικά τη δυνατότητα ορισμού συναρτήσεων που παρέχεται από μια γλώσσα διαδικαστικού προγραμματισμού (procedural programming) κατ′ αναλογία με τη δυνατότητα ορισμού κλάσεων του αντικειμενοστραφούς προγραμματισμού.

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

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

class A{ int data;
  public:
    A(int dt=0) : data(dt) { cout << "Creating an A" << endl; }
    A(const A& a) { data = a.data; cout << "Creating an A by Copying" << endl; }
    ~A() { cout << "Destroying an A with data" <<  data << endl; }
    void inc() { data++; } };

int main(){ A a1; A a2(10); A a3(a2); A a4 = a2; A& a5 = a1; a5 = a2;
            a1.inc(); a2.inc(); a3.inc(); a4.inc(); a5.inc(); }
```

   Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσής του. Για κάθε εκτύπωση, αναφέρατε από ποιά κλήση συνάρτησης προέκυψε.

2. Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης του παρακάτω προγράμματος C++:

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

class A{ int data;
  public:
    A(int dt=0) : data(dt) { cout << "Creating an A" << endl; }
    A(const A& a) { data = a.data; cout << "Creating an A by Copying" << endl; }
    ~A() { cout << "Destroying an A with data " <<  data << endl; }
    void inc(){ data++; }
    A  naughty1() {  return *this; }
    A&  naughty2() { return *this; } };

class BigA{ A a; A& ra; A* pa;
  public:
    BigA(const A& a1, A& a2, A* pa3=NULL) :
        a(a1), ra(a2), pa(pa3)  { cout << "Constructing a BigA" << endl ;}
    A& inc() { a.inc(); ra.inc(); if(pa != NULL) pa->inc(); return a;} };

int main(){ A data; BigA big_a(data,data,&data);
            big_a.inc();
            data.inc(); big_a.inc().naughty1();
            data.inc(); big_a.inc().naughty2();
            big_a.inc().naughty1().naughty2();
            big_a.inc().naughty2().naughty1(); }
```

3. Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης του παρακάτω προγράμματος C++:

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

class drink{
  protected : int value;
  public:
   drink(int val=5) : value(val) { cout << "offering a drink" << endl; }
   virtual void serve() = 0;
   virtual int get_value() { return value; } };

class hot : public drink{
   public:
     hot() { cout << " a hot one " << endl;}
     void serve() { cout << "Careful it's hot! " << "Please pay "
                         << get_value() << " euros " << endl; } };

class milk : public hot{ int value;
  public:
    milk(int val=3) : value(val) { cout << " milk!! " << endl; }
    int get_value() { return value; }
    void serve() { value += drink::value; hot::serve(); } };

class chocolate : public hot{
   public:
     chocolate() { cout << " a chocolate! " << endl; } };


class cold : public drink{
   public:
     cold() { cout << " a cold one " << endl;}
     void serve() { cout << "Please pay " << get_value() << " euros " << endl; } };

class refreshment : public cold{
  public:
     refreshment() { cout << " a refreshment!" << endl;} };

class coctail : public cold{ int value;
  public:
   coctail(int val=5) : value(val) { cout << " a coctail!" << endl;}
   int get_value() { return value; }
   void serve() { value += drink::value; cold::serve(); } };

class bar{ drink** drks;
   public:
     bar() { drks = new drink*[6];
             drks[0] = new hot(); drks[1] = new cold();
             drks[2] = new milk(); drks[3] = new chocolate();
             drks[4] = new refreshment(); drks[5] = new coctail(); }
     void serve() { for (int i=0; i< 6; i++) drks[i]->serve(); } };
```

```
int main(){ bar br; br.serve(); }
```

4. Να υλοποιηθεί σε C++ ένα μέρος της λειτουργικότητας ενός ηλεκτρονικού καταστήματος. Συγκεκριμένα, υλοποιήστε τους πελάτες (`customers`) και τα προϊόντα (`items`) του. Τα προϊόντα συνδέονται ανά ζεύγη, οπότε το ζεύγος προσφέρεται σε ιδιαίτερη (μειωμένη) τιμή. Επίσης, προκειμένου να παρέχει τη δυνατότητα προτάσεων αγοράς (recommendations) προς τους πελάτες του, διατηρεί συστάδες συναφών προϊόντων (`item_clusters`). Αυτές έχουν προκύψει από το ιστορικό των αγορών με κοινό σημείο τους πελάτες που τα αγόρασαν. Οι προτάσεις προς ένα πελάτη προκύπτουν ταιριάζοντας τα προϊόντα που έχει αγοράσει ο πελάτης στο παρελθόν με κάποια από τις συστάδες. Οι συστάδες θεωρούνται γνωστές και ο αλγόριθμος κατασκευής τους καθώς και εκείνος του ταιριάσματος ενός συνόλου προϊόντων με μια συστάδα δεν αφορά την υλοποίηση.

   Η κλάση "Πελάτης"

   - έχει μια ταυτότητα (`id`)
   - έχει ένα καλάθι προϊόντων που έχει επιλέξει για αγορά (`items_basket`)
   - έχει ένα ποσό που πρέπει να πληρώσει και αντιστοιχεί στην αξία των προϊόντων του καλαθιού (`total_cost`)
   - έχει ένα σύνολο από προϊόντα που έχει αγοράσει στο παρελθόν (`past_items`)

   Η κλάση "Πελάτης" χαρακτηρίζεται από την εξής συμπεριφορά:

   - κατά την κατασκευή του, εκχωρείται η ταυτότητά του και το ιστορικό των προϊόντων που έχει αγοράσει. Το καλάθι του είναι κενό και η αντίστοιχη αξία ίση με μηδέν.
   - επιλέγει ένα προϊόν για αγορά. Τότε αυτό εκτυπώνεται στην οθόνη και, αν υπάρχει απόθεμα από αυτό, προστίθεται στο καλάθι του και αυξάνεται το ποσόν που είναι να πληρώσει κατά την αξία του προϊόντος. Αν επιλέξει το προϊόν και το προϊόν-ταίρι, αφού γίνει έλεγχος αποθέματος και για το ταίρι, προστίθενται και τα δύο στο καλάθι και η συνολική αξία που προστίθεται αφορά την μειωμένη τιμή.
   - πραγματοποιεί αγορά, μηδενίζοντας το ποσό που πρέπει να πληρώσει, αδειάζοντας το καλάθι και προσθέτοντας τα περιεχόμενα του καλαθιού στο σύνολο των προϊόντων που έχει αγοράσει. Πραγματοποιείται αγορά των προϊόντων του καλαθιού με βάση την ταυτότητά του.
   - με βάση τις συστάδες προϊόντων του καταστήματος, του γίνονται προτάσεις αγοράς σα σύνολο προϊόντων. Οι προτάσεις αυτές παρέχονται από τη χρήση μιας συνάρτησης που ενεργεί με βάση ένα σύνολο προϊόντων και ένα σύνολο συστάδων.

   Η κλάση "Προϊόν"

   - έχει μια ταυτότητα (`id`)
   - έχει μια ονομασία (`name`)
   - έχει μια τιμή (`price`)
   - έχει τη ποσότητα του προϊόντος σε απόθεμα (`stock`)
   - έχει ένα ταίρι-προϊόν. Το ταίρι αυτό περιέχει ένα άλλο προϊόν καθώς και τη τιμή που κοστίζει το ζεύγος αγοράς (`partner`)
   - έχει το σύνολο των ταυτοτήτων των πελατών που το έχουν αγοράσει στο παρελθόν (`customers`)

   Η κλάση "Προϊόν" χαρακτηρίζεται από την εξής συμπεριφορά:

   - κατά την κατασκευή του, εκχωρείται η ταυτότητά του, η ονομασία του, η τιμή του, το αρχικό απόθεμά του και το ιστορικό των ταυτοτήτων των πελατών που το έχουν αγοράσει. Το ταίρι-προϊόν είναι κενό.

- ανατίθεται το ταίρι-προϊόν. Κατά την ανάθεση, ενημερώνεται και το αντίστοιχο πεδίο του προϊόντος του ταιριού.
- ελέγχεται αν υπάρχει απόθεμα από το προϊόν.
- εκτυπώνεται στην οθόνη, εκτυπώνοντας την ονομασία του, την τιμή του, το απόθεμά του και το ταίρι-προϊόν (όνομα του άλλου προϊόντος και μειωμένη συνολική τιμή).
- πραγματοποιείται αγορά του προϊόντος, μειώνοντας κατά ένα το απόθεμα —υποθέτουμε ότι η ποσότητα από το κάθε προϊόν που μπορεί να αγοράσει ένας πελάτης είναι μόνο μια μονάδα— και ενημερώνονται οι ταυτότητες των πελατών που το έχουν αγοράσει με την ταυτότητά του πελάτη που το αγοράζει.

Τα προϊόντα υποδιαιρούνται στις παρακάτω κατηγορίες:

"βιβλία" (`books`): αυτά επιπλέον έχουν συγγραφέα (`author`) και έτος έκδοσης (`year`)

"DVDs" (`DVDs`): αυτά επιπλέον έχουν έτος παραγωγής (`year`)

"παιχνίδια και ηλεκτρονικά" (`toys_and_games`): αυτά επιπλέον έχουν εύρος ηλικιών για τα οποία είναι κατάλληλα (`ages`)

Για τα παραπάνω, η αρχικοποίηση και η εκτύπωση στην οθόνη λαμβάνει υπόψη της και τα παραπάνω στοιχεία.

Οι συστάδες προϊόντων είναι σύνολα προϊόντων.

Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις, ορίζοντας νέες όπου δείχνει ενδεικνυόμενο.
Σημείωση: Όπου θεωρείτε απαραίτητο, κάνετε παραδοχές στις προδιαγραφές, τεκμηριώνοντάς τις. Θεωρήστε δεδομένο έναν τύπο λίστας με συναρτήσεις προσθήκης στοιχείου και συνένωσης λιστών. Θεωρήστε δεδομένη τη συνάρτηση παροχής προτάσεων προϊόντων με βάση ένα σύνολο προϊόντων και ένα σύνολο συστάδων.
(Για ενημέρωσή σας, η παραπάνω περιγραφή ανταποκρίνεται στις απαιτήσεις του αλγόριθμου Item-to-Item Collaborative Filtering.)