ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 28 Ιανουαρίου 2020

1. (α′) Στον αντικειμενοστραφή προγραμματισμό συναντάμε τις σχέσεις *"is a"* και *"has a"* μεταξύ κλάσεων. Τι εννοούμε με τις σχέσεις αυτές; Δώστε παράδειγμα κλάσεων σε C++ που να σχετίζονται μεταξύ τους με αυτές τις σχέσεις.

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

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

class Data { int data;
    public:
      Data(int i = 0) : data(i) { cout << "Data constructed " << endl; }
      ~Data() { cout << "Data to be destructed with: " << data << endl; }
      void inc() { data++; }
      void print() { cout << "Printing Data with: " << data << endl; } };

class A{ Data* data;
   public:
      A(const Data& dt) { cout << "A constructed " << endl; data = new Data(dt); }
      A(const A& a) { cout << "A constructed by CP " << endl;
                      data = new Data(*(a.data)); }
      virtual ~A() { cout << "A to be destructed with Data: " << endl; data->print();
                     delete data; }
      void inc() { data->inc(); }
      virtual void print() { cout << "Printing a A with: " << endl; data->print();}};

class B : public A{ Data& data;
   public:
      B(Data& dt) : data(dt), A(dt) { cout << "B constructed " << endl; }
      ~B() { cout << "B to be destructed with Data: " << endl; data.print(); }
      void inc() { A::inc(); data.inc(); }
      void print() { A::print(); cout << "Printing a B with: " << endl; data.print();}};

class C : public B{ Data data;
   public:
      C(Data& dt) : data(dt), B(dt) { cout << "C constructed " << endl;}
      ~C() { cout << "C to be destructed with Data: " <<  endl; data.print(); }
      void inc() { B::inc(); data.inc(); }
      void print() { B::print(); cout << "Printing a C with: " << endl; data.print();}};

class Fat1{ A data1; A& data2;
  public:
    Fat1(A& a): data1(a), data2(a) { cout << " Fat1 just constructed " << endl; }
    ~Fat1() { cout << "A Fat1 to be destructed " << endl; }
    void inc() { data1.inc(); data2.inc(); }
    void print() { data1.print(); data2.print(); } };

class Fat2{ B data1; B& data2;
  public:
    Fat2(B& b): data1(b), data2(b) { cout << " Fat2 just constructed " << endl; }
```

```
    ~Fat2() { cout << "A Fat2 to be destructed " << endl; }
    void inc() { data1.inc(); data2.inc(); }
    void print() { data1.print(); data2.print(); } };

int main(){  Data dt(200); A a(dt); B b(dt); C c(dt);
   a.inc(); b.inc(); c.inc(); a.print(); b.print(); c.print();

   Fat1 f1(a); Fat2 f2(b);
   f1.inc(); f2.inc(); f1.print(); f2.print();

   A* p1 = &a; A* p2 = &b; A* p3 =&c;
   p1->inc(); p2->inc(); p3->inc(); p1->print(); p2->print(); p3->print(); }
```

Δώστε τα αποτελέσματα της εκτέλεσης του προγράμματος, αιτιολογώντας την απάντησή σας.

2. Ας θεωρήσουμε έναν *φοιτητή* που παρακολουθεί ένα πρόγραμμα προπτυχιακών σπουδών ενός πανεπιστημιακού τμήματος. Το πρόγραμμα αυτό αποτελείται από *μαθήματα*. Τα μαθήματα μπορεί να είναι είτε *υποχρεωτικά μαθήματα* είτε *προαιρετικά μαθήματα* του τμήματος. Ο αριθμός των υποχρεωτικών μαθημάτων είναι καθορισμένος εξ αρχής. Επίσης, υπάρχουν και *ελεύθερα μαθήματα* τα οποία προσφέρονται από άλλα τμήματα. Κάθε μάθημα έχει ένα όνομα καθώς και το πλήθος των ECTS που το χαρακτηρίζει.

   Κατηγορία υποχρεωτικών μαθημάτων είναι τα *υποχρεωτικά μαθήματα κατεύθυνσης* που πρέπει υποχρεωτικά να παρακολουθήσει επιτυχώς ένας φοιτητής στην περίπτωση που αποφασίσει να ακολουθήσει την κατεύθυνση αυτή. Έστω ότι οι κατευθύνσεις έχουν τιμές `'A'` και `'B'`. Ένα υποχρεωτικό μάθημα κατεύθυνσης περιέχει και την κατεύθυνση στην οποία ανήκει. Ο αριθμός των υποχρεωτικών μαθημάτων κατεύθυνσης είναι καθορισμένος εξ αρχής και κοινός και για τις δύο κατευθύνσεις.

   Κατηγορία προαιρετικών μαθημάτων είναι τα *βασικά μαθήματα ειδίκευσης*. Οι ειδικεύσεις έχουν τιμές "E1", "E2", "E3", "E4", "E5", "E6". Ένα βασικό μάθημα ειδίκευσης περιέχει και την ειδίκευση της οποίας είναι βασικό (μόνο μία). Για να γίνει κατοχύρωση ειδίκευσης (`specialization_ok`), ένας φοιτητής πρέπει να παρακολουθήσει επιτυχώς τουλάχιστον έναν συγκεκριμένο αριθμό μαθημάτων από την ειδίκευση αυτή. Ο αριθμός αυτός είναι κοινός για τα μαθήματα όλων των ειδικεύσεων και μπορεί να μεταβάλλεται μέσω μιας διαδικασίας αλλαγής τιμής (`change_min`).

   Τα ελεύθερα μαθήματα συνοδεύονται κι από το όνομα του τμήματος που τα προσφέρει.

   Κάθε φοιτητής έχει έναν αριθμό μητρώου, έχει δηλώσει κατεύθυνση και έχει παρακολουθήσει επιτυχώς ένα σύνολο μαθημάτων. Για τον φοιτητή, μπορεί να ζητηθεί να εκτυπωθεί η πληροφορία αυτή (`print`). Στην περίπτωση αυτή ζητείται να εκτυπώνεται όλη η πληροφορία των μαθημάτων. Επίσης, μπορεί να ζητηθεί να εκτιμηθεί αν έχει κατοχυρώσει ειδίκευση (`specialization_ok`), αν έχει κατοχυρώσει συγκεκριμένη ειδίκευση (`specialization_ok`) και αν έχει ολοκληρώσει τις υποχρεώσεις λήψης πτυχίου (`finished`). Για το τελευταίο, πρέπει να έχει παρακολουθήσει επιτυχώς όλα τα υποχρεωτικά μαθήματα, όλα τα υποχρεωτικά μαθήματα κατεύθυνσης και να έχει συμπληρώσει τον απαιτούμενο αριθμό ECTS. Επίσης, υπάρχει ένα μέγιστο στο πλήθος των ECTS από ελεύθερα μαθήματα που μπορεί να ληφθεί υπόψη για την ολοκλήρωση των υποχρεώσεων λήψης πτυχίου. Θεωρήστε ότι ο φοιτητής αρχικοποιείται με τα στοιχεία του καθώς και τα μαθήματα που έχει παρακολουθήσει επιτυχώς.

   Κάθε μια από τις παραπάνω οντότητες, μπορεί να αρχικοποιηθεί, παρέχοντας τις τιμές των επί μέρους στοιχείων της και να εκτυπωθεί, εκτυπώνοντας όλες τις πληροφορίες που σχετίζονται με αυτήν (`print`).

   Υλοποιήστε την παραπάνω πληροφορία σε C++, υλοποιώντας κατάλληλες κλάσεις, ορίζοντας τα μέλη-δεδομένα που χρειάζονται και συναρτήσεις-μέλη που υλοποιούν την παραπάνω συμπεριφορά. (*Σημείωση:* Δεν χρειάζεται υλοποίηση συνάρτησης main).

   ΠΡΟΣΟΧΗ -προς αποφυγή παρεξηγήσεων: Το ερώτημα είναι εμπνευσμένο από το ΠΠΣ του Τμήματός μας αλλά οι προδιαγραφές του δεν ταυτίζονται με αυτό!