ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 1 Σεπτεμβρίου 2008

1. (α′) Στον αντικειμενοστραφή προγραμματισμό μιλάμε για "επαναχρησιμοποίηση" κλάσεων μέσω "σύνθεσης" ή μέσω "κληρονομικότητας". Πού αναφέρεται ο καθένας όρος από αυτούς; Δώστε ένα παράδειγμα κλάσεων C++ που συνδυάζονται μέσω σύνθεσης και ένα παράδειγμα κλάσεων C++ που συνδυάζονται μέσω κληρονομικότητας.

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

```
#include<iostream>

using namespace std;

class A{
   void set_data3(int i) { data3 = i; }
   int data1;
  protected:
   void set_data1(int i) { data1 = i; }
   int data2;
  public:
   A(int i = 0) : data1(i), data2(i), data3(i) {}
   ~A() { cout << "Destroying a A with data " << data1 << "  " <<
                  data2 << "  " << data3 << endl; }
   int data3;
   void set_data2(int i) {data2 = i; } };

class B : public A{
   int data1;
  public :
   B(int i = 0) : A(i), data1(i) {}
   ~B() { cout << "Destroying a B with data " << data1 << endl; }
   void set_data1(int i) { data1 = i; }
   void process(int i) { DATA = i; SETDATA(i); } };

int main(){
    B b; b.process(500);
  return 0; }
```

   Με ποιά από τα ονόματα των μελών-δεδομένων των παραπάνω κλάσεων είναι επιτρεπτό να αντικατασταθεί η λέξη `DATA`; Ποιές είναι οι επιτρεπτές κλήσεις συναρτήσεων-μελών που μπορούν να αντικαταστήσουν την έκφραση `SETDATA(i)`; Για κάθε συνδυασμό τέτοιων αντικαταστάσεων, αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης του προγράμματος.

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

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

class Block{ int data;
  public:
    Block(int dt = 0) : data(dt)
```

```
        {cout << "Creating a Block with " << data << endl;}
    Block(const Block& bk)
        { cout << "Creating a Block by Copying with " << bk.data << endl;
          data = bk.data; }
    ~Block() {cout << "Destroying a Block with " << data << endl;}
    int get() {return data;}
    void set(int dt) {data = dt;} };

class B{ Block& data1; Block  data2; Block* data3;
   public:
     B(Block& blk1, Block& blk2, Block* pblk3 = NULL) : data1(blk1), data2(blk2)
        { cout << "Creating a B " << endl;
          if (pblk3 != NULL)
                data3 = new Block(*pblk3);
              else
                data3 = NULL; }
     B(const B& b) : data1(b.data1), data2(b.data2)
        { cout << "Creating a B by copying " << endl;
          if (b.data3 != NULL)
                data3 = new Block(*(b.data3));
              else
                data3 = NULL; }
     ~B() {cout << "Destroying a B" << endl;
           if (data3 != NULL) delete data3;}
     void set(int dt) {data1.set(dt); data2.set(dt);
                       if (data3 != NULL) data3->set(dt);}
     void print() {cout <<  "The B data are " << data1.get() << "  "
                        << data2.get() << "  " << data3->get() << endl;} };

B& first(B& b) {b.set(10); b.print(); return b;}
B second(B& b) {b.set(5); b.print(); return b;}
B third(B b) {b.set(1000); b.print(); return b;}

int main(){Block block; B b1(block,block,&block);
           b1.print(); first(b1); second(b1); third(b1); block.set(300); return 0;}
```

3. 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{
  protected : int strength;
  public :
   Actor(int s=10) : strength(s) {cout << "Creating an Actor" << endl;}
   Actor(const Actor& actor) {cout << "Creating a Copy of an Actor" << endl;
                              strength = actor.strength;}
   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 sun!" << endl; tiring();} };

class ActorSingerDancer : public ActorDancer{
  public:
    ActorSingerDancer(int astrength=40, int dstrength=50) :
                      ActorDancer(astrength, dstrength)
         {cout << "Creating an Actor who Sings and Dances" << endl;}
    virtual void act() {ActorDancer::act(); cout << "I am singing !" << endl;
                        tiring(); Actor::tiring(); } };

class Musical{
   ActorSingerDancer& starring;
   ActorDancer& dancer1; ActorDancer& dancer2; ActorDancer& dancer3;
   Actor& actor1; Actor& actor2; Actor& actor3;
  public :
    Musical(ActorSingerDancer& s, ActorDancer& d1, ActorDancer& d2,
            ActorDancer& d3, Actor& a1, Actor& a2, Actor& a3) :
            starring(s), dancer1(d1), dancer2(d2), dancer3(d3),
            actor1(a1), actor2(a2), actor3(a3)
       {cout << "The show starts!" << endl;
        scene(s,d1,d2); scene(d1,d2,d3); scene(a1,a2,a3);}
    ~Musical() {cout << "The End" << endl;}
//    void scene(Actor a1, Actor a2, Actor a3)
//    void scene(Actor& a1, Actor& a2, Actor& a3)
               { a1.act(); a2.act(); a3.act();} };

int main(){ ActorSingerDancer MS; Actor PB, SS, CF;
            ActorDancer AD1, AD2, AD3;
            Musical mamma_mia(MS, AD1, AD2, AD3, PB, SS, CF);
    return 0; }
```

4. Να υλοποιηθεί σε C++ ένα μέρος της λειτουργικότητας ενός συστήματος αγγελιών εύρεσης εργασίας. Συγκεκριμένα, υλοποιήστε τις αγγελίες (posts) και τους ενδιαφερόμενους (users). Επίσης, υλοποιήστε και την ιεραρχία επαγγελμάτων (ενδεικτικό μέρος της). Το σύστημα αποτελείται από ένα σύνολο ενδιαφερόμενων και ένα σύνολο αγγελιών. Ένας ενδιαφερόμενος μπορεί να επιλέξει το πολύ MAX αριθμό αγγελιών. Η επιλογή είναι επιτρεπτή αν τα χαρακτηριστικά του ενδιαφερόμενου ταιριάζουν με τις ανάγκες της αγγελίας. Μια αγγελία είναι διαθέσιμη για επιλογή, αν ο αριθμός των ενδιαφερομένων για αυτή δεν έχει ξεπεράσει ένα μέγιστο (MAX_CUST). Η έννοια του επαγγέλματος των ενδιαφερομένων

αποτελεί μια ιεραρχία, όπου κάθε ειδίκευση στο επάγγελμα συνοδεύεται από συγκεκριμένα χαρακτηριστικά. Για παράδειγμα, ο "Προγραμματιστής εφαρμογών του web" (`web_programmer`) είναι εξειδίκευση του "Προγραμματιστή" (`programmer`) ο οποίος με τη σειρά του είναι εξειδίκευση του "Πληροφορικού" (`computer_scientist`) Ένας "Πληροφορικός" έχει γνώσεις από "υλικό", "λογισμικό", "μαθηματικά" και "φυσική". Ένας "Προγραμματιστής" έχει γνώσεις από κάποιες γλώσσες προγραμματισμού. Ένας "Προγραμματιστής εφαρμογών του web" είναι εξοικειωμένος με κάποια περιβάλλοντα και γλώσσες εφαρμογών του web. Τόσο για τους "Προγραμματιστές" όσο και για τους "Προγραμματιστές εφαρμογών του web" οι γνώσεις δεν είναι προκαθορισμένες.

Η κλάση "Αγγελία"

- έχει μια ταυτότητα (`id`)
- έχει έναν τίτλο (`title`)
- έχει ένα σύνολο γνώσεων που απαιτούνται από τους ενδιαφερόμενους (`skills`)
- έχει ένα ελάχιστο και ένα μέγιστο ηλικιών που, προαιρετικά, έχουν τιμές (`min_age`), (`max_age`)
- έχει, προαιρετικά, ένα ποσό που αντιστοιχεί στο μισθό (`salary`)
- έχει μια ένδειξη αν η αγγελία είναι διαθέσιμη για να την επιλέξει κάποιος ενδιαφερόμενος (`availability`)
- έχει ένα σύνολο ενδιαφερομένων που την έχουν επιλέξει (`users`).

Η κλάση "Αγγελία" χαρακτηρίζεται από την εξής συμπεριφορά:

- κατά την κατασκευή της, εκχωρείται η ταυτότητά της, ο τίτλος της, το σύνολο των γνώσεων που απαιτούνται από τους ενδιαφερόμενους και οι ηλικίες και ο μισθός. Στην περίπτωση που τα τελευταία δεν προσδιορίζονται, δίδεται η τιμή 0. Αρχικά, η αγγελία είναι διαθέσιμη και το σύνολο των ενδιαφερόμενων που την επέλεξαν είναι κενό.
- επιλέγεται από έναν ενδιαφερόμενο αν είναι διαθέσιμη και τα χαρακτηριστικά του ενδιαφερόμενου ταιριάζουν με τις απαιτήσεις της (γνώσεις, ηλικία, μισθό). Τότε, προστίθεται ο ενδιαφερόμενος στο σύνολο των ενδιαφερόμενων. Γίνεται έλεγχος, αν μετά τη προσθήκη, η αγγελία παύει να είναι διαθέσιμη οπότε η διαθεσιμότητα αλλάζει.

Η κλάση "Ενδιαφερόμενος"

- έχει μια ταυτότητα (`id`)
- έχει ένα επάγγελμα (`profession`)
- έχει μια ηλικία (`age`)
- έχει μια μισθολογική προτίμηση (`salary`)
- έχει ένα σύνολο αγγελιών τις οποίες έχει επιλέξει (`posts`)

Η κλάση "Ενδιαφερόμενος" χαρακτηρίζεται από την εξής συμπεριφορά:

- κατά την κατασκευή του, εκχωρείται η ταυτότητά του, το επάγγελμά του, η ηλικία του και η μισθολογική προτίμησή του. Αρχικά, δεν έχει επιλέξει καμιά αγγελία.
- επιλέγει αγγελία, αν δεν έχει φτάσει στο όριο, επιλέγοντας την και προσθέτοντας την αγγελία στο σύνολο των αγγελιών που έχει επιλέξει —αν η ενέργεια της επιλογής έγινε επιτυχώς από την αγγελία—.

Τα επαγγέλματα, τώρα, έχουν ένα σύνολο γνώσεων που προσφέρουν. Κάθε εξειδίκευση επαγγέλματος, πέρα από τις γνώσεις που προσφέρουν οι γενικότερες κατηγορίες, προσφέρει και κάποιες επιμέρους. Κάποιες κατηγορίες επαγγελμάτων προσφέρουν συγκεκριμένες γνώσεις, για κάποιες άλλες όμως, οι επιπλέον γνώσεις προσδιορίζονται κατά περίπτωση.

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.