ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 12 Σεπτεμβρίου 2007

1. (α′) Αναφέρετε επιγραμματικά τα βασικά χαρακτηριστικά του διαδικαστικού προγραμματισμού (procedural programming), της αφαίρεσης στα δεδομένα (data abstraction) και του αντικειμενοστραφούς προγραμματισμού.

   (β′) Έστω ότι έχουμε τις παρακάτω C++ κλάσεις:

```
class C1{ int data;
  public:
   XXX C1(int dat=0) : data(dat) {}
   YYY ~C1(){ cout << "Deleting a C1" << endl; }
   ZZZ void print() { cout << data << endl; } };

class C2 : public C1{ int* data;
  public:
   WWW C2(int dat=0) { if (dat==0) data = NULL; else data = new int(dat); }
   VVV ~C2() { cout << "Deleting C2 in all cases" << endl; delete data; }
   UUU void print() { if (data!=NULL) cout << *data << endl; }};
```

   Αν υποθέσουμε ότι τα `XXX`, `YYY`, `ZZZ`, `WWW`, `VVV` και `UUU` μπορούν να αντικατασταθούν είτε με το κενό είτε με τη λέξη `virtual`, ποιες αντικαταστάσεις δεν μπορούν να γίνουν και ποιές ενδείκνυται να γίνουν; Αιτιολογείστε την απάντησή σας.

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 dat=0) : data(dat)
          { cout << "Building a Block with data :" << dat << endl ;}
    int inc() { return data++; }
    void add(int i) { data = data + i; }
    void print() { cout << "Block data is :" << data << endl; } };

class B{ Block b1; Block& b2; Block* b3;
  public:
   B(Block& blk1, Block& blk2 , Block* pblk=NULL) :
     b1(blk1), b2(blk2), b3(pblk)
     { cout << "I just constructed a B" << endl; }
   void inc() { b1.inc(); b2.inc(); if (b3!=NULL) b3->inc(); }
   void add(int i) { b1.add(i); if (b3!=NULL) b3 ->add(i); }
   B naughty() { return *this; }
   void print() {b1.print(); b2.print(); if (b3!=NULL) b3->print();} };

main(){ Block block(10); B b(block, block, &block); b.print();
    b.inc(); b.print(); block.print();
    b.add(3); b.print(); block.print();
    b.add(block.inc()); b.print(); block.print();
    b.naughty().inc(); b.print(); block.print(); }
```

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

```
#include<iostream>
using namespace std;
enum Efrt {zero, little, enough, hard};
enum WTh {positive, negative};


class Person{
  protected: WTh way_of_thinking; Efrt effort;
  public:
   Person(Efrt efrt, WTh wth=positive) :
          effort(efrt), way_of_thinking(wth) {}
   virtual bool is_good() = 0;
   virtual void earn_living() = 0;
   virtual void socialize(Person* pp)
     { way_of_thinking = pp->way_of_thinking; };
   void improve(Person* pp){
        if (effort <= pp->effort) effort = pp->effort;
            else pp->effort = effort; }};


class GoodPerson : public Person{
  public:
   GoodPerson(Efrt efrt): Person(efrt) {}
   bool is_good() {return true;}
   void socialize(Person* pp){
        if (pp->is_good()) improve(pp);
            else pp->Person::socialize(this); }
   void earn_living(){
             switch(effort){
             case hard : cout << "I earn a lot of money" << endl; break;
             case enough : cout << "I earn enough money" << endl; break;
             case little : cout << "I earn little money" << endl; break;
             default: cout << "I need a good influence" << endl; } } };


class WickedPerson : public Person{
  public:
   WickedPerson() : Person(zero, negative) {}
   bool is_good() { return false;}
   void earn_living() {
             switch(way_of_thinking){
             case positive :
                  cout << "I'll make an honest effort" << endl; break;
             default : cout << "You'd better not ask me" << endl; } } };

main(){
   GoodPerson gp1(zero); GoodPerson gp2(little);
   GoodPerson gp3(enough); GoodPerson gp4(hard);
   WickedPerson wp;
```

```
Person* persons[] = {&gp1, &gp2, &gp3, &gp4, &wp};
for(int i=0; i<5; i++) persons[i]->earn_living();

for(int i=0; i<4; i++) persons[i+1]->socialize(persons[i]);
for(int i=0; i<5; i++) persons[i]->earn_living();

for(int i=4; i>0; i--) persons[i-1]->socialize(persons[i]);
for(int i=0; i<5; i++) persons[i]->earn_living(); }
```

4. Να υλοποιηθεί σε C++ μια προσομοίωση εκλογικού κέντρου. Το εκλογικό κέντρο αποτελείται από δέκα εκλογικά τμήματα. Στο εκλογικό κέντρο προσέρχονται εκλογείς για να ψηφίσουν, κάνοντας μια επιλογή από ένα (συγκεκριμένο) πλήθος κομμάτων, λευκό ή άκυρο (για λόγους απλότητας, θεωρήστε τις δυο τελευταίες επιλογές σαν δυο ακόμα κόμματα). Το εκλογικό κέντρο επιτρέπει την είσοδο εκλογέων μέχρι τις 19:00 και κατόπιν κλείνει. Τα εκλογικά τμήματα, όμως, εξυπηρετούν όλους τους εκλογείς που περιμένουν να ψηφίσουν. Οι εκλογείς αναπαρίστανται από το επίθετό τους και την πρόθεση ψήφου τους. Για την προσομοίωση να χρησιμοποιηθούν οι κλάσεις "Εκλογικό Κέντρο" και "Εκλογικό Τμήματα".

   Η κλάση "Εκλογικό Κέντρο"

   - έχει ένα ρολόι για να δείχνει την ώρα (`roloi`)
   - έχει έναν αριθμό που αντιστοιχεί στους ψηφίζοντες στο κέντρο (`psifizontes`)
   - έχει έναν αριθμό που αντιστοιχεί στους ψηφίσαντες στο κέντρο (`psifisantes`)
   - φυλάσσει το πλήθος των ψήφων που έχουν πάρει τα κόμματα (`apotelesmata`)
   - έχει δέκα εκλογικά τμήματα (`tmimata`)
   - έχει μια ένδειξη αν είναι ανοικτό ή κλειστό (`open`)

   Η κλάση "Εκλογικό Κέντρο" χαρακτηρίζεται από την εξής συμπεριφορά:

   - αρχικά, είναι ανοικτό, η ώρα είναι 6:00, κανείς δεν έχει ψηφίσει, καταχωρεί τον αριθμό των ψηφιζόντων του και καταχωρεί τα εύρη των επιθέτων που εξυπηρετούνται από τα τμήματά του.
   - όταν εισέρχεται ένας νέος ψηφοφόρος, αν το κέντρο είναι ανοικτό, πάει στο κατάλληλο τμήμα και προστίθεται να ψηφίσει. Το τμήμα επιλέγεται με βάση το επίθετο του ψηφοφόρου, λαμβάνοντας υπόψη ότι ένα τμήμα εξυπηρετεί ψηφοφόρους από ένα συγκεκριμένο εύρος επιθέτων.
   - τα αποτελέσματα των κομμάτων προκύπτουν από τα επιμέρους αποτελέσματα των τμημάτων
   - το πλήθος των ψηφισάντων στο κέντρο προκύπτει από τους ψηφίσαντες στα τμήματα
   - αν είναι κλειστό, και δεν υπάρχει κάποιο τμήμα στο οποίο να περιμένουν ψηφοφόροι, ανακοινώνονται τα αποτελέσματά του (κόμματα, άκυρα, λευκά, αποχή)
   - αν η ώρα είναι 19:00, το κέντρο κλείνει

   Η κλάση "Εκλογικό Τμήμα"

   - έχει το εύρος επιθέτων που εξυπηρετεί (`arxika`)
   - έχει ένα σύνολο ψηφοφόρων που περιμένουν να ψηφίσουν (`oura_psifoforon`)
   - έχει έναν μετρητή για το πόσοι ψηφοφόροι έχουν ψηφίσει (`psifisantes`)
   - φυλάσσει τα μέχρι τώρα αποτελέσματα (`apotelesmata`)

   Η κλάση "Εκλογικό Τμήμα" χαρακτηρίζεται από την εξής συμπεριφορά:

   - κατά την κατασκευή του, δεν έχει ψηφίσει κανείς και δεν περιμένει κανείς να ψηφίσει.
   - καταχωρείται το εύρος των επιθέτων που εξυπηρετεί.

- ένας νέος ψηφοφόρος προστίθεται να ψηφίσει, περιμένοντας στο τέλος των ψηφοφόρων που περιμένουν να ψηφίσουν, αν το επίθετό του ανήκει στο εύρος επιθέτων που εξυπηρετούνται από το τμήμα.
- ο πρώτος ψηφοφόρος που περιμένει να ψηφίσει, ψηφίζει αυξάνοντας τους ψηφίσαντες του τμήματος κατά ένα καθώς και τα αποτελέσματα του κόμματος της επιλογής του. Επίσης, αφαιρείται από την ουρά αναμονής των ψηφοφόρων.

Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις, ορίζοντας νέες όπου δείχνει ενδεικνυόμενο.
Σημείωση: Όπου θεωρείτε απαραίτητο, κάνετε παραδοχές στις προδιαγραφές, τεκμηριώνοντάς τις. Θεωρήστε δεδομένη τη συνάρτηση ελέγχου, αν ένα επίθετο βρίσκεται σε ένα εύρος επιθέτων.