ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 1 Οκτωβρίου 2015

1. (α′) Ποιός μηχανισμός πρέπει να υπάρχει προκειμένου να προσφερθεί η διάκριση μεταξύ private και public ονομάτων στις γλώσσες όπως η C++; Τι πετυχαίνουμε με τη διάκριση αυτή και πού αυτό είναι χρήσιμο στην ανάπτυξη κώδικα;

   (β′) 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 i = 10) : data(i)
          { cout << "I just created a Block " << endl; }
      ~Block() { cout << "I will destroy a Block with " << data << endl; }
      void inc() { data++; } };

class A{ Block& block1; Block block2;
     public :
      A(Block& blk) : block1(blk),block2(blk)
       { cout << "I just created an A " << endl; }
      A(const A& a): block1(a.block1), block2(a.block2) {
cout << "I just created an A by copying but I will also do bad things" << endl;
            block1.inc(); block2.inc(); }
      ~A() { cout << "I will destroy an A " << endl; }
      void inc() { block1.inc(); block2.inc(); } };

class Fat{ A  a; A& ra; A*  pa;
      public:
       Fat(A& da) : a(da),ra(da) { pa = new A(da);
                       cout << "Fat just created !" << endl;}
       ~Fat() {  delete pa;
                 cout << "Fat to be destroyed !" << endl; }
       void inc() { a.inc(); ra.inc(); pa->inc(); } };

int main(){ Block block; A a(block); Fat fat(a);
            fat.inc();
   return 0; }
```

2. Δίδεται το παρακάτω πρόγραμμα C++:

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

class Book {
   protected:
      const int pages; int time_per_page;
   public:
      Book(int p = 100, int tpp = 10) : pages(p), time_per_page(tpp)
           { cout << "New Book with: " << pages << " and "
                  << time_per_page << endl; }
```

```
      ~Book() { cout << " Book to be destroyed! "<< endl; }
//      int get_time_needed() { return pages * time_per_page;}
//      virtual int get_time_needed() { return pages * time_per_page;}
//      int modify_fatigue(int init_fatig) { return init_fatig *= 2;}
//      virtual int modify_fatigue(int init_fatig) { return init_fatig *= 2;}
      void announcement() { cout << "I am reading a Book" << endl;} };

class Novel : public Book {
    public :
      Novel(int p):Book(p) { cout << " New Novel just created! " << endl; }
      ~Novel() { cout << " Novel to be destroyed! " << endl; } };

class Science : public Book {
    public :
      Science(int p):Book(p) { cout << " New Science just created! " << endl; }
      ~Science() { cout << " Science to be destroyed! " << endl; }
      int get_time_needed(){ time_per_page*=10; return Book::get_time_needed();}
      int modify_fatigue(int init_fatig) { return init_fatig *= 10;} };

class Cartoon : public Book {
    public :
      Cartoon(int p):Book(p) { cout << " New Cartoon just created! " << endl; }
      ~Cartoon() { cout << " Cartoon to be destroyed! " << endl; }
      int modify_fatigue(int init_fatig) { return init_fatig /= 5;} };

class Reader{
      int available_time; int fatigue;
      Book& book1; Book& book2; Book& book3;
      void read(Book& book) { book.announcement();
                              available_time -= book.get_time_needed();
                              fatigue = book.modify_fatigue(fatigue);}
   public:
      Reader(int at, int f, Book& b1, Book& b2, Book& b3)
            : available_time(at), fatigue(f), book1(b1), book2(b2), book3(b3)
            { cout << " A New Reader has been created! " << endl;}
      ~Reader()
            { cout << " A Reader to be destroyed with! " << available_time
                   << " and " << fatigue << endl;}
       void read() { read(book1);
                           cout << available_time << "   " << fatigue << endl;
                     if (available_time>= fatigue)
                         { read(book2);
                           cout << available_time << "   " << fatigue << endl;
                           if(available_time >= fatigue)
                              { read(book3);
                              cout << available_time << "   " << fatigue << endl;
                                   cout << " I finished reading! " << endl;}
                              else
                               cout <<" I give up!" << endl;}
                           else
                     cout << "Too many books!" << endl; } };
```

```
int main(){

   Novel the_great_gatsby(20); Science calculus(50); Cartoon asterix(100);
   Reader r1(1000,600,the_great_gatsby,calculus,asterix);
   Reader r2(200000,50,calculus,asterix,the_great_gatsby);
   r1.read(); r2.read();

   return 0; }
```

Αποσχολιάστε σχολιασμένες γραμμές με τρόπο ώστε το πρόγραμμα να μεταγλωττίζεται και, αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσής του, κάθε φορά.

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία προσομοίωση εισαγωγής επισκεπτών σε ένα κτήριο δημόσιας υπηρεσίας. Προκειμένου να γίνει η υλοποίηση αυτή θεωρούμε τις κλάσεις: "επισκέπτης" (`Visitor`), "κτήριο" (`Building`), "όροφος" (`Floor`), "χώρος εισόδου" (`Entrance`) και "γραφείο" (`Office`) .

   Η κλάση "επισκέπτης":

   - έχει ένα κτήριο που θέλει να επισκεφτεί (`building`)
   - έχει ένα αριθμό γραφείου, του γραφείου που επιθυμεί να επισκεφτεί (`no_office`)

   Η κλάση "επισκέπτης" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά ανατίθεται το κτηρίο και ο αριθμός του γραφείου που θέλει να επισκεφτεί.
   - Μπαίνει στο κτήριο, αν του επιτρέπεται (`enter`).

   Η κλάση "κτήριο":

   - έχει τέσσερεις ορόφους (`floors`)
   - έχει έναν πίνακα αντιστοίχησης αριθμών γραφείων και αριθμών ορόφων στον οποίο βρίσκεται το καθένα (`table`)

   Η κλάση "κτηριο" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Ένας επισκέπτης μπαίνει στο κτήριο, βρίσκοντας τον αριθμό του ορόφου στον οποίο ανήκει ο αριθμός του γραφείου που θέλει να επισκεφτεί και μπαίνοντας στον όροφο αυτόν, αν αυτό επιτρέπεται (`enter`).

   Η κλάση "όροφος":

   - έχει έναν χώρο εισόδου (`entance`)
   - έχει δέκα γραφεία (`offices`)

   Η κλάση "οροφος" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά αναθέτει τις ενδείξεις των γραφείων του στο χώρο εισόδου του
   - Ένας επισκέπτης μπαίνει στον όροφο, μπαίνοντας στο χώρο εισόδου του, αν του επιτρέπεται (`enter`).

Η κλάση "χώρος εισόδου":

- έχει μια μέγιστη χωρητικότητα (`capacity`)
- έχει ένα μετρητή για το πλήθος των επισκεπτών που βρίσκονται μέσα σε αυτόν τη στιγμή αυτή (`no_of_visitors`)
- έχει ενδείξεις των γραφείων του ορόφου στον οποίο ανήκει (`p_offices`)

Η κλάση "χώρος εισόδου" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά είναι άδειος και του ανατίθεται η μέγιστη χωρητικότητά του
- Ένας επισκέπτης μπαίνει στον χώρο εισόδου, αν δεν υπερβαίνεται η μέγιστη χωρητικότητά του. Αν του επιτρέπεται να μπει στο γραφείο που επιθυμεί, μπαίνει στο γραφείο. Διαφορετικά παραμένει στο χώρο εισόδου, αυξάνοντας το μετρητή του πλήθους των επισκεπτών του (`enter`).

Η κλάση "γραφείο":

- έχει μια μέγιστη χωρητικότητα(`capacity`)
- έχει ένα μετρητή για το πλήθος των επισκεπτών που βρίσκονται μέσα σε αυτό τη στιγμή αυτή (`no_of_visitors`)

Η κλάση "γραφείο" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά είναι άδειο και του ανατίθεται η μέγιστη χωρητικότητά του
- Ένας επισκέπτης μπαίνει στο γραφείο αν δεν υπερβαίνεται η μέγιστη χωρητικότητά του, αυξάνοντας το μετρητή του πλήθους των επισκεπτών του (`enter`).

Υλοποιήστε τα παραπάνω μέσω καταλλήλων κλάσεων, ορίζοντας μέλη-δεδομένα που χρειάζονται και συναρτήσεις-μέλη που υλοποιούν την παραπάνω συμπεριφορά.

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.