ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 6 Φεβρουαρίου 2015

1. (α′) Αναφέρετε τρεις λέξεις ή εκφράσεις, διαφορετικές από "κλάση" και "αντικείμενο", που συναντά κανείς στο πλαίσιο του αντικειμενοστραφούς προγραμματισμού και δώστε μια επιγραμματική περιγραφή του νοήματος τους.

   (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(const Block& blk) { data = blk.data;
            cout << "I just created a Block by copying with data:" <<
            data << 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() { cout << "I will destroy an A " << endl; }
      void inc() { block1.inc(); block2.inc(); }
      A& operator=(const A& a)
         { if (this != &a)
              { block1 = a.block1;
                cout << "Not a wise assignment!"<< endl;}
            return *this; } };

class Fat{ A  a; A& ra; A*  pa;
      public:
       Fat(A& da) : a(da),ra(da) { a = 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;}
//      virtual int modify_fatigue(int init_fatig) { return init_fatig *= 2;}
//      int modify_fatigue(int init_fatig) { return init_fatig *= 2;}
      virtual void announcement() = 0; };

class Novel : public Book {
    public :
      Novel(int p):Book(p) { cout << " New Novel just created! " << endl; }
      ~Novel() { cout << " Novel to be destroyed! " << endl; }
     int modify_fatigue(int init_fatig) { return init_fatig;}
     void announcement() { cout << " I am reading a Novel!" << 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;}
      void announcement() { cout << " I am reading Science!" << endl; } };

class Cartoon : public Book {
    public :
      Cartoon(int p):Book(p) { cout << " New Cartoon just created! " << endl; }
      ~Cartoon() { cout << " Cartoon to be destroyed! " << endl; }
      int get_time_needed(){ time_per_page/=2; return Book::get_time_needed();}
      int modify_fatigue(int init_fatig) { return init_fatig /= 5;}
      void announcement() { cout <<  " What a funny Cartoon!" << endl; } };

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);
                     if (available_time>= fatigue) { read(book2);
                                                     if(available_time >= fatigue)
                           { read(book3); 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 της φόρτωσης επιβατηγών πλοίων. Προκειμένου να γίνει η υλοποίηση αυτή θεωρούμε τις κλάσεις: "επιβάτης" (`Passenger`), "επιβατικό αυτοκίνητο" (`Car`), "φορτηγό αυτοκίνητο" (`Truck`) και "επιβατηγό πλοίο" (`Ferry_boat`).

   Η κλάση "επιβάτης":

   - έχει έναν μοναδικό αριθμό ταυτότητας (`id`)
   - έχει ένα πλοίο στο οποίο θα επιβιβαστεί (`boat`)
   - έχει μια ώρα άφιξης στην προβλήτα που είναι αγκυροβολημένο το πλοίο (`arrival_time`)
   - είτε έχει είτε δεν έχει αγοράσει εισιτήριο για το πλοίο (`ticket`)

   Η κλάση "επιβάτης" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά ανατίθεται ο αριθμός ταυτότητάς του, το πλοίο, η ώρα άφιξης στην προβλήτα καθώς και το εάν έχει αγοράσει εισιτήριο ή όχι.
   - Επιβιβάζεται ένας επιβάτης στο πλοίο, εάν η ώρα άφιξής του είναι μικρότερη από την ώρα αναχώρησης του πλοίου και εάν με την επιβίβασή του δεν υπερβαίνουμε το μέγιστο αριθμό επιβατών του πλοίου. Τότε, στην περίπτωση που δεν έχει εισιτήριο, αγοράζει εισιτήριο και αυξάνεται ο μετρητής επιβιβασμένων επιβατών του πλοίου, (`board`).

   Η κλάση "επιβατικό αυτοκίνητο":

   - έχει έναν μοναδικό αριθμό κυκλοφορίας (`id`)
   - έχει ένα πλοίο στο οποίο θα επιβιβαστεί (`boat`)
   - έχει μια ώρα άφιξης στην προβλήτα που είναι αγκυροβολημένο το πλοίο (`arrival_time`)
   - έχει ένα σύνολο επιβατών που μεταφέρει (`passengers`), το πολύ επτά, καθώς και το πλήθος τους (`passengers_size`). Οι επιβάτες του θεωρούμε ότι έχουν αγοράσει εισιτήρια. Η ώρα άφιξης των επιβατών του είναι η ίδια με με την ώρα άφιξης του αυτοκινήτου.

   Η κλάση "επιβατικό αυτοκίνητο" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά καταχωρείται ο αριθμός κυκλοφορίας του, το πλοίο, η ώρα άφιξης στην προβλήτα καθώς και οι επιβάτες που μεταφέρει.
   - Το αυτοκίνητο επιβιβάζεται στο πλοίο, εάν η ώρα άφιξής του είναι μικρότερη από την ώρα αναχώρησης του πλοίου και εάν με την επιβίβασή του δεν υπερβαίνουμε το μέγιστο αριθμό οχημάτων του πλοίου. Τότε, αυξάνεται ο μετρητής επιβιβασμένων οχημάτων του πλοίου και επιβιβάζουμε τους επιβάτες του (`board`).

   Η κλάση "φορτηγό αυτοκίνητο":

   - έχει έναν μοναδικό αριθμό κυκλοφορίας (`id`)
   - έχει ένα πλοίο στο οποίο θα επιβιβαστεί (`boat`)
   - έχει μια ώρα άφιξης στην προβλήτα που είναι αγκυροβολημένο το πλοίο (`arrival_time`)

- έχει έναν μόνο επιβάτη, τον οδηγό, που ενδέχεται και να μην έχει εισιτήριο. Η ώρα άφιξης του οδηγού του είναι η ίδια με με την ώρα άφιξης του φορτηγού (`driver`).

Η κλάση "φορτηγό αυτοκίνητο" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά καταχωρείται ο αριθμός κυκλοφορίας του, το πλοίο, η ώρα άφιξης στην προβλήτα καθώς και ο οδηγός του.
- Το φορτηγό επιβιβάζεται στο πλοίο, εάν η ώρα άφιξής του είναι μικρότερη από την ώρα αναχώρησης του πλοίου και εάν με την επιβίβασή του δεν υπερβαίνουμε το μέγιστο αριθμό οχημάτων του πλοίου. Τότε, αυξάνεται ο μετρητής επιβιβασμένων οχημάτων του πλοίου και επιβιβάζουμε τον οδηγό του (`board`).

Η κλάση "επιβατηγό πλοίο":

- έχει έναν μοναδικό αριθμό νηολογίου (`id`)
- έχει μια ώρα αναχώρησης (`departure_time`)
- έχει έναν μετρητή επιβιβασμένων επιβατών (`no_passengers`)
- έχει ένα μέγιστο αριθμό επιβατών (`max_no_passengers`)
- έχει έναν μετρητή επιβιβασμένων οχημάτων (`no_vehicles`)
- έχει ένα μέγιστο αριθμό οχημάτων (`max_no_vehicles`)
- έχει μια ουρά επιβατών που περιμένουν να μπουν στο καράβι (`passenger_queue`)
- έχει μια ουρά οχημάτων (επιβατικών αυτοκινήτων ή φορτηγών) που περιμένουν να μπουν στο καράβι (`vehicle_queue`)

Η κλάση "επιβατηγό πλοίο" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά καταχωρείται ο αριθμός νηολογίου του, η ώρα αναχώρησης, ο μέγιστος αριθμός επιβατών και ο μέγιστος αριθμός οχημάτων. Αρχικά το πλοίο είναι άδειο και οι ουρές κενές.
- Προστίθεται ένας επιβάτης στην ουρά επιβατών, τοποθετώντας τον στο τέλος της ουράς. Στην περίπτωση που η ώρα άφιξης του επιβάτη είναι μεγαλύτερη από την ώρα αναχώρησης του πλοίου, δε μπορεί να προστεθεί στην ουρά (`passenger_arrive`).
- Αφαιρείται ένας επιβάτης από την ουρά επιβατών, διαγράφοντάς τον από την ουρά και επιβιβάζοντάς τον στο πλοίο (`passenger_enter`).
- Προστίθεται ένα επιβατικό αυτοκίνητο ή φορτηγό στην ουρά οχημάτων, τοποθετώντας το στο τέλος της ουράς. Στην περίπτωση που η ώρα άφιξής του είναι μεγαλύτερη από την ώρα αναχώρησης του πλοίου, δε μπορεί να προστεθεί στην ουρά (`vehicle_arrive`).
- Αφαιρείται ένα επιβατικό αυτοκίνητο ή φορτηγό από την ουρά οχημάτων, διαγράφοντάς το από την ουρά και επιβιβάζοντάς το στο πλοίο (`vehicle_enter`).

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

Σημείωση: Θεωρήστε δεδομένο έναν τύπο ουράς με τις λειτουργίες που χρειάζεστε.

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.