ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 9 Φεβρουαρίου 2016

1. (α′) Τι εννοούμε με τον όρο "εγκλωβισμός" (encapsulation); Γιατί είναι κρίσιμος για τον αντικειμενοστραφή προγραμματισμό; Δώστε ένα παράδειγμα C++ κώδικα στο οποίο υπάρχει χρήση εγκλωβισμού.

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

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

class A{int var;
 public:
    A(int i=10):var(i){
                cout << "I just constructed an A with " << var << endl;}
    ~A(){cout << "An A will be destroyed with " << var << endl;}
    void bla(){cout << "I am A:bla() " << endl; var++;}
    void bloo() {cout << "I am A::blo()" << endl; bla();} };

class B: public A{int var;
 public:
    B(int i=100):var(i){
                 cout << "I just constructed a B with " << var << endl;}
    ~B(){cout << "A B will be destroyed with " << var << endl;}
    void bla(){cout << "I am B:bla() " << endl; var++;} };

class C: public A{int var;
 public:
    C(int i=200):var(i){
                 cout << "I just constructed a C with " << var << endl;}
    ~C(){cout << "A C will be destroyed with " << var << endl;}
    virtual void bla(){cout << "I am C:bla() " << endl; var++;} };

class D: public B{int var;
 public:
    D(int i=1000):var(i){
                  cout << "I just constructed a D with " << var << endl;}
    ~D(){cout << "A D will be destroyed with " << var << endl;} };

class E: public C{int var;
 public:
    E(int i=2000):var(i){
                  cout << "I just constructed an E with " << var << endl;}
    ~E(){cout << "An E will be destroyed with " << var << endl;}
    void bla(){ cout << "I am E:bla() " << endl; var++;} };

class F: public C{int var;
 public:
    F(int i=2222):var(i){
                  cout << "I just constructed a F with " << var << endl;}
    ~F(){cout << "A F will be destroyed with " << var << endl;} };
```

```
int main(){ A* pa; B* pb; C* pc; A a; D d; E e; F f;
    a.bla(); a.bloo();  d.bla(); d.bloo();
    e.bla(); e.bloo();  f.bla(); f.bloo();

    pa=&a; pa->bla(); pa->bloo();
    pa=&d; pa->bla(); pa->bloo();
    pa=&e; pa->bla(); pa->bloo();
    pa=&f; pa->bla(); pa->bloo();

    pb=&a; pb->bla(); pb->bloo();
    pb=&d; pb->bla(); pb->bloo();

    pc=&d; pc->bla(); pc->bloo();
    pc=&e; pc->bla(); pc->bloo();
    pc=&f; pc->bla(); pc->bloo(); }
```

Το πρόγραμμα αυτό δεν μεταγλωττίζεται. Εξηγήστε γιατί. Χωρίς να αλλάξετε τους ορισμούς των κλάσεων, σχολιάστε τις ελάχιστες γραμμές ώστε αυτό να μεταγλωττίζεται. Στη συνέχεια, αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσής του.

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

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

class A{int i;
   public:
      A(int ii=10):i(ii)
                {cout << "I just constructed an A with" << i << endl;}
      A(const A& a):i(a.i)
                {cout << "I just constructed an A by COPYING with "
                      << i << endl;}
      ~A() {cout << "I shall destroy an A with " << i << endl;}
  int inc() {return ++i;}
  int get_data() const {return i;} };

class B{A& a;
   public:
      B(A& aa):a(aa)
             {cout << "I just constructed a B with "
             << a.get_data() << endl;}
      B(const B& b):a(b.a)
             {cout << "I just constructed a B by COPYING with "
              << a.get_data() << endl;}
      ~B() {cout << "I shall destroy a B" << endl;}
  int inc() {return a.inc();}
  int get_data() const {return a.get_data();} };

class C{ A a; B b;
   public:
      C(const A& aa, B& bb):a(aa),b(bb)
         {cout << "I just constructed a C" << endl;}
```

```
      C(const C& c):a(c.a),b(c.b)
         {cout << "I just constructed a C by COPYING" << endl;}
      ~C() {cout << "I shall destroy a C" << endl;}
      void inc() {a.inc(); b.inc();}
      B get_b() {return b;} };

class D{A* a; B b; C& c;
   public:
      D(A& aa, const B& bb, C& cc):a(&aa),b(bb),c(cc)
          {cout << "I just constructed a D" << endl;}
      D(const D& d):a(d.a),b(d.b),c(d.c)
          {cout << "I just constructed a D by COPYING" << endl;}
      ~D() {cout << "I shall destroy a D " << endl;}
      void inc() {a->inc(); b.inc(), c.inc();}
      B get_b() {return b;} };

int main(){A a(20); B b(a); C c(a,b); D d(a,b,c);
           a.inc(); b.inc(); c.inc(); d.inc();
           c.get_b().inc(); d.get_b().inc(); }
```

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία απλοϊκή προσομοίωση της εισόδου και εξόδου αυτοκινήτων σε ένα γκαράζ. Στο γκαράζ θεωρούμε ότι σταθμεύουν τόσο (επιβατικά) αυτοκίνητα όσο και φορτηγά, σε τρεις ορόφους. Το γκαράζ έχει μια μέγιστη χωρητικότητα για στάθμευση αυτοκινήτων και μια μέγιστη χωρητικότητα για στάθμευση φορτηγών, ισοκατανεμημένες στους τρεις ορόφους. Προκειμένου να εισέλθει στο γκαράζ ένα αυτοκίνητο ή ένα φορτηγό, προστίθεται το τέλος μιας ουράς εισόδου που οδηγεί στο γκαράζ. Το πρώτο αυτοκίνητο ή φορτηγό της ουράς, αφαιρείται από την ουρά και μπαίνει στο γκαράζ, σταθμεύοντας στον πρώτο όροφο που θα βρεθεί ελεύθερη θέση. Βγαίνει από το γκαράζ το τελευταίο αυτοκίνητο ή φορτηγό (ό,τι είναι αυτό) που στάθμευσε σε έναν όροφο.

   Προκειμένου να γίνει η υλοποίηση αυτή θεωρούμε τις κλάσεις: "επιβατικό αυτοκίνητο" (`Car`), "φορτηγό" (`Truck`), "γκαράζ" (Garage), "όροφος" (Floor) και "ουράΕισόδου" (EntranceQueue).

   Η κλάση "επιβατικό αυτοκίνητο":

   - έχει έναν όροφο στον οποίο έχει σταθμεύσει (`floor`)

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

   - Αρχικά, δεν έχει σταθμεύσει σε κάποιον όροφο.
   - Το αυτοκίνητο εισέρχεται σε μια ουρά εισόδου γκαράζ, εισάγοντας στην ουρά εισόδου το αυτοκίνητο (`enter`).
   - Το αυτοκίνητο σταθμεύει σε έναν όροφο, εάν επιτρέπεται από τον αντίστοιχο μετρητή, αναθέτοντας τον όροφο αυτόν σαν τιμή του ορόφου του και αυξάνοντας τον μετρητή αυτοκινήτων (`park`).
   - Το αυτοκίνητο βγαίνει, αναθέτοντας στον όροφό του μηδενική τιμή και μειώνοντας τον μετρητή αυτοκινήτων του ορόφου στον οποίο είχε σταθμεύσει (`exit`).
   - Το αυτοκίνητο επιστρέφει τον όροφο στον οποίο έχει σταθμεύσει (`get_floor`).
   - Εκτυπώνεται μήνυμα πληρότητας αυτοκινήτων `"Car places are full"` (`print_wait_message`).

   Η κλάση "φορτηγό αυτοκίνητο":

   - έχει έναν όροφο στον οποίο έχει σταθμεύσει (`floor`)

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

   - Αρχικά, δεν έχει σταθμεύσει σε κάποιον όροφο.

- Το φορτηγό εισέρχεται σε μια ουρά εισόδου γκαράζ, εισάγοντας στην ουρά εισόδου το φορτηγό (`enter`).
- Το φορτηγό σταθμεύει σε έναν όροφο, εάν επιτρέπεται από τον αντίστοιχο μετρητή, αναθέτοντας τον όροφο αυτόν σαν τιμή του ορόφου του και αυξάνοντας τον μετρητή φορτηγών (`park`)
- Το φορτηγό βγαίνει, αναθέτοντας στον όροφό του μηδενική τιμή και μειώνοντας τον μετρητή φορτηγών του ορόφου στον οποίο είχε σταθμεύσει (`exit`).
- Το φορτηγό επιστρέφει τον όροφο στον οποίο έχει σταθμεύσει (`get_floor`).
- Εκτυπώνεται μήνυμα πληρότητας φορτηγών `"Truck places are full"` (`print_wait_message`).

Η κλάση "γκαράζ":

- έχει τρεις ορόφους (`floor1`, `floor2`, `floor3`)
- έχει ένα μέγιστο πλήθος αυτοκινήτων (χωρητικότητα αυτοκινήτων) που επιτρέπεται να είναι σταθμευμένα στο γκαράζ (`car_max`)
- έχει ένα μέγιστο πλήθος φορτηγών (χωρητικότητα φορτηγών) που επιτρέπεται να είναι σταθμευμένα στο γκαράζ (`truck_max`)

Η κλάση "γκαράζ" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά ανατίθενται οι χωρητικότητες αυτοκινήτων και φορτηγών.
- Ένα αυτοκίνητο ή φορτηγό μπαίνει σε αυτό, μπαίνοντας στον πρώτο όροφο που χωράει. Αν δεν υπάρχει τέτοιος, εκτυπώνεται το σχετικό μήνυμα πληρότητας (`enter`).
- Ένα αυτοκίνητο ή φορτηγό βγαίνει από αυτό, βγαίνοντας από τον όροφο που είναι σταθμευμένο (`exit`).

Η κλάση "όροφος":

- έχει ένα μετρητή πλήθους αυτοκινήτων που είναι σταθμευμένα στον όροφο (`car_counter`)
- έχει ένα μετρητή πλήθους φορτηγών που είναι σταθμευμένα στον όροφο (`truck_counter`)
- έχει ένα σύνολο αυτοκινήτων ή φορτηγών που είναι σταθμευμένα σε αυτόν πλήθους το πολύ όσο το άθροισμα των χωρητικοτήτων του (`parked_Car_Trucks`)

Η κλάση "όροφος" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά είναι άδειος.
- Ένα αυτοκίνητο ή φορτηγό μπαίνει σε αυτόν, αν μπορεί να σταθμεύσει, και προστίθεται στο σύνολο των σταθμευμένων οχημάτων (`enter`).
- Αφαιρείται το τελευταίο σταθμευμένο αυτοκίνητο ή φορτηγό από τον όροφο, μηδενίζοντας τη θέση του και βγάζοντάς το από τον όροφο (`exit`).

Η κλάση "ουράΕισόδου":

- έχει ένα γκαράζ στο οποίο οδηγεί (`garage`)
- έχει μια ουρά οχημάτων (`queue_Car_Trucks`)

Η κλάση "ουράΕισόδου" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά της ανατίθεται το γκαράζ.
- Ένα αυτοκίνητο ή φορτηγό μπαίνει σε αυτή, προστιθέμενο στο τέλος της ουράς της (`enter`).
- Το πρώτο αυτοκίνητο ή φορτηγό βγαίνει από αυτή, αφαιρούμενο από την ουρά της και μπαίνοντας στο γκαράζ (`exit`).

Υλοποιήστε τις κατάλληλες κλάσεις, ορίζοντας μέλη-δεδομένα που χρειάζονται και συναρτήσεις-μέλη που υλοποιούν την παραπάνω συμπεριφορά. Θεωρήστε δεδομένο έναν τύπο ουράς με τις λειτουργίες που χρειάζεστε. Μην υλοποιήσετε τη συνάρτηση `main` της προσομοίωσης.