ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10:ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 9 Σεπτεμβρίου 2013

1. (α′) i. Τι επιτυγχάνουμε με το μηχανισμό του εγκλωβισμού (encapsulation) στον αντικειμενοστραφή προγραμματισμό; 1) ταχύτητα στην εκτέλεση 2) αφαίρεση δεδομένων 3) πολυμορφισμό 4) κληρονομικότητα 5) ορισμό αντικειμένων

   ii. Πού είναι χρήσιμος ο μηχανισμός του εγκλωβισμού στον προγραμματισμό, γενικότερα; 1) διευκολύνει την ανάπτυξη εκτενούς κώδικα 2) διευκολύνει την υποστήριξη κάθε γλώσσας προγραμματισμού 3) διευκολύνει την υλοποίηση απλών αριθμητικών προβλημάτων 4) απαιτείται από το λειτουργικό σύστημα 5) τίποτα από τα προηγούμενα

   iii. Στις γλώσσες αντικειμενοστραφούς προγραμματισμού, κατά τον ορισμό μιας κλάσης χωρίζουμε το δημόσιο μέρος της από το ιδιωτικό. Ποιός μηχανισμός απαιτείται από τη γλώσσα προκειμένου να επιτύχουμε τη διάκριση αυτή; 1) ελέγχου τύπων 2) υποστήριξης τύπων ορισμένων από το χρήστη 3) εγκλωβισμού 4) εμβέλειας χωρίς απόκρυψη πληροφορίας 5) υποστήριξης ορισμού αντικειμένων

   iv. Τι εννοούμε με τον όρο "εμβέλεια" ονομάτων; 1) τη δήλωση ονόματος συνάρτησης 2) τη δήλωση ονόματος μεταβλητής 3) το μέρος του προγράμματος που ένα όνομα είναι συσχετισμένο με ένα νόημα 4) δεν υπάρχει τέτοιος όρος 5) τίποτα από τα εναλλακτικά

   v. Ένας τρόπος ορισμού εμβέλειας στη C++ είναι 1) η κλάση 2) οι χώροι ονομάτων 3) τα blocks 4) όλα τα προηγούμενα 5) τίποτα από τα προηγούμενα

   (β′) Δίδεται το παρακάτω πρόγραμμα C++

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

class A {
      int data;
    public:
      A(int i = 0) : data(i)
        { cout << "I am constructing an A with " << i << endl; }
      ~A() { cout << "Destructing an A with " << data << endl; }
      virtual void bla() { cout << "Calling A::bla" << endl; data++; }
      void fun() { cout << "Calling A::fun" << endl; bla(); }
      virtual void bloo() = 0; };

class B : public A {
      int data;
    public:
      B(int i = 100) : data(i)
        { cout << "I am constructing a B with " << i << endl; }
      ~B() { cout << "Destructing an B with" << data << endl; }
      void bla() { cout << "Calling B::bla" << endl; data++;}
      void bloo() { cout << "Calling B::bloo " << endl; A::bla(); } };

// void foo(A param){
// void foo(A& param){
// void foo(B param){
// void foo(B& param){
     param.bla(); param.bloo(); param.fun(); }
```

```
int main(){
//  A data; foo(data);
//  B data; foo(data);
    return 0;  }
```

Αποσχολιάζοντας 2 γραμμές τη φορά από το παραπάνω πρόγραμμα, ποιές μπορούν να αποσχολιαστούν για να μπορεί να εκτελεστεί; Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης.

2. Δίδεται το παρακάτω πρόγραμμα C++:

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

class A1 { int data;
   public:
     A1(int i = 10 ) : data(i)
        { cout << "I am constructing an A1 with: " << i << endl; }
     A1(const A1& a1) : data(a1.data)
        { cout << "I am copy constructing an A1" << endl; }
     ~A1() { cout << "I am destroying an A1 with: " << data << endl; }
     A1& operator=(const A1& a1)
        { cout << "I am performing an A1 assignment" << endl;
          if (this != &a1) data = a1.data;
          return *this ;}
     void change() { data = data * 10; } };

class A2 { int data;
   public:
     A2(int i = 20 ) : data(i)
        { cout << "I am constructing an A2 with: " << i << endl; }
     A2(const A2& a2) : data(a2.data)
        { cout << "I am copy constructing an A2" << endl; }
     ~A2() { cout << "I am destroying an A2 with: " << data << endl; }
     A2& operator=(const A2& a2)
        { cout << "I am performing a stupid assignment on A2s" << endl;
          if (this != &a2) {data = a2.data; data++;}
          return *this ;}
     void change() { data = data * 20; } };

class A3 {
   public:
     A3() { cout << "I am constructing an A3" << endl; }
     A3(const A3& a3)
        { cout << "I am copy constructing an A3" << endl; }
     ~A3() { cout << "I am destroying an A3" << endl; }
     void change() { cout << "Nothing to change" << endl; } };

class A { A1 a1; A2 a2; A3 a3;
   public:
     A() { cout << "I am constructing an A" << endl; }
     ~A() { cout << "I am destroying an A" << endl; }
     void change() { a1.change(); a2.change(); a3.change(); } };
```

```
class BigA { A data1; A& data2;
     public:
      BigA(A& a): data1(a), data2(a)
          { cout << "I just constructed a BigA" << endl; }
     ~BigA() { cout << "I am destroying a BigA" << endl; }
      A& get(int index) { if (index == 1) return data1;
                          else return data2; } };

// BigA volta(BigA& biga)
// BigA& volta(BigA& biga)
           { cout << "Volta ta data?" << endl;
             return biga; }

int main() {
    A first; A second(first); A third = first;
    second = third;

    BigA biga(first);
    volta(biga).get(2).change();

    return 0; }
```

Αποσχολιάζοντας μια από τις δύο σχολιασμένες γραμμές τη φορά, δώστε το αποτέλεσμα της εκτέλεσης, αιτιολογώντας την απάντησή σας.

3. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία σκηνή επιδείξεων παγοδρομιών. Στη σκηνή μπαίνουν παγοδρόμοι που τοποθετούνται ανά ζευγάρια διαφορετικού φύλου. Υπάρχει ένας μέγιστος αριθμός ζευγαριών που μπορούν να σχηματιστούν. Οι επιπλέον παγοδρόμοι που μπαίνουν σχηματίζουν δύο αλυσίδες: μία οι άνδρες και μία οι γυναίκες. Όταν, και αν, οι αλυσίδες αυτές φτάσουν και οι δύο σε ένα μέγιστο μήκος, συνενώνονται και φτιάχνουν μια μεγάλη, διπλάσιου μήκους, όπου οι άνδρες και οι γυναίκες τοποθετούνται εναλλάξ. Δεν επιτρέπεται να μπουν επιπλέον παγοδρόμοι. Οι παγοδρόμοι αναχωρούν, πρώτα εκείνοι της αλυσίδας και κατόπιν τα ζευγάρια.

   Για την υλοποίση της παραπάνω συμπεριφοράς να χρησιμοποιηθούν οι κλάσεις "Σκηνή" (`Scene`), "Παγοδρόμος" (`IceSkater`), "Ζευγάρι" (`Couple`), "ΑλυσίδαΙδιουΦύλου" (`SSChain`) και "Αλυσίδα" (`Chain`).

   Η κλάση "Σκηνή":

   - έχει μια μέγιστη χωρητικότητα παγοδρόμων (`capacity`)
   - έχει ένα σύνολο ζευγαριών (`couples`)
   - έχει μια ένδειξη σε αλυσίδα ανδρών (`menchain`)
   - έχει μια ένδειξη σε αλυσίδα γυναικών (`womenchain`)
   - έχει μια ένδειξη σε αλυσίδα παγοδρόμων (`chain`)

   Η κλάση "Σκηνή" χαρακτηρίζεται από την εξής συμπεριφορά:

   - Αρχικά, καταχωρείται η χωρητικότητά της και είναι άδεια από παγοδρόμους.
   - Ένας παγοδρόμος μπαίνει στη σκηνή, αν χωράει. Αν δεν χωράει, εκτυπώνεται σχετικό μήνυμα. Αν χωράει, γίνεται δοκιμή να τοποθετηθεί σε ζευγάρι. Αν αποτύχει η τοποθέτηση αυτή, τοποθετείται στην αλυσίδα του φύλου του, αν μπορεί (`enter`)
   - Γίνεται συνένωση των αλυσίδων ιδίου φύλου, αφαιρώντας τους παγοδρόμους από τις αλυσίδες ιδίου φύλου, έναν από κάθε μία τη φορά, και βάζοντάς τους στην αλυσίδα παγοδρόμων (`populate`)

- Γίνεται λήξη της σκηνής, κάνοντας αναχώρηση των παγοδρόμων, πρώτα των αλυσίδων και μετά των ζευγαριών (`scene_end`)

Η κλάση "Παγοδρόμος":

- έχει καταχωρημένο το φύλο του, `M` ή `F`, (`sex`)

Η κλάση "Παγοδρόμος" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά, καταχωρείται το φύλο του.
- Ένας παγοδρόμος αναχωρεί εκτυπώνοντας το μήνυμα `"I am gliding out"` και το φύλο του (`exit`)

Η κλάση "Ζευγάρι":

- έχει δύο ενδείξεις σε παγοδρόμους (`left`, `right`)

Η κλάση "Ζευγάρι" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά, είναι κενή από παγοδρόμους.
- Ένας παγοδρόμος γίνεται μέλος στο ζευγάρι, αν του είναι δυνατό, δηλαδή είτε είναι κενές και οι δύο θέσεις είτε μόνο η μια θέση είναι κατειλημμένη και ο παγοδρόμος που την καταλαμβάνει είναι του άλλου φύλου. Αν η εισαγωγή είναι επιτυχής, καταχωρείται στο ζευγάρι μια ένδειξη προς αυτό τον παγοδρόμο (`enter`)

Η κλάση "ΑλυσίδαΙδιουΦύλου":

- έχει μια μέγιστη χωρητικότητα παγοδρόμων (`capacity`)
- έχει ένα σύνολο παγοδρόμων (`ice_skaters`)

Η κλάση "ΑλυσίδαΙδιουΦύλου" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά καταχωρείται η χωρητικότητά της και το σύνολο των παγοδρόμων είναι κενό. Η χωρητικότητα υπολογίζεται με βάση τη χωρητικότητα της σκηνής και το πλήθος των ζευγαριών των παγοδρόμων.
- Ένας παγοδρόμος μπαίνει στην αλυσίδα, αν είναι του φύλου του και δεν υπερβαίνει τη χωρητικότητά της. Αν δεν μπορεί να μπει, εκτυπώνεται σχετικό μήνυμα (`enter`)

Η κλάση "Αλυσίδα":

- έχει ένα σύνολο παγοδρόμων (`ice_skaters`)

Η κλάση "Αλυσίδα" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά το σύνολο των παγοδρόμων είναι κενό.
- Ένας παγοδρόμος μπαίνει στην αλυσίδα σε άκρη που υπάρχει παγοδρόμος του άλλου φύλου. Αν δεν μπορεί να μπει, εκτυπώνεται σχετικό μήνυμα (`enter`)

Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις. Θεωρήστε δεδομένο έναν τύπο "λίστας" όπως σας βολεύει, απλώς τεκμηριώστε τον. Όπου θεωρείτε απαραίτητο, κάνετε παραδοχές στις προδιαγραφές, τεκμηριώνοντάς τις.

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.