ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Κ10: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις 28 Σεπτεμβρίου 2006

1. (α′) Τι εννοούμε με τον όρο "προγραμματισμός με στοιχεία" (modular programming) και τι με τον όρο "αφαίρεση δεδομένων" (data abstraction); Ποιός μηχανισμός είναι απαραίτητος για την επίτευξή τους;

   (β′) Έστω ότι έχουμε το παρακάτω πρόγραμμα:

```
#include <iostream.h>
class A{ int i;
 public:
   A(int ival=0): i(ival)
     {cout << "I just constructed an A with: " << ival << endl;}
   A(const A& a)
     {cout << "I just constructed an A by copying" << endl; i = a.i;}
   A& operator=(const A& a)
      {cout << "Do I really change anything?" << endl; return *this; }
   ~A() {cout << "I am destructing an A with: " << i << endl;} };

A& bla(A& pa)
{ A blaa(5); A blab(pa); A blac = pa;  A& blad = pa;  blad = blaa;
  return XXXX; }

main(){ A a; bla(a); }
```

   Με ποιό/ά από τα `blaa, blab, blac, blad` είναι ασφαλές να αντικαταστήσουμε το `XXXX` και γιατί; Αντικαθιστώντας, πείτε τι εκτυπώνεται κατά την εκτέλεση του προγράμματος και δώστε το πρότυπο της συνάρτησης (`function prototype`) που καλείται κατά την εκτέλεση κάθε εντολής της `main` και της `bla`.

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

```
#include <iostream.h>
class Block{ int data;
 public:
   Block(int datav = 0) : data(datav)
        { cout << "Constructing a Block with " << datav << endl; }
   ~Block(){ cout << "Destructing a Block with: " << data << endl; }
   void double_it() { data = data * 2 ; }
   void print() { cout << data << endl; } };

class B{ Block data1; Block* data2; Block* data3;
 public:
  B() : data2(NULL),data3(NULL) { cout << "Constructing a B" << endl; }
  B(int value1, Block* value2) : data1(value1)
        { cout << "Constructing a B with given values "  << endl;
                  data2 = new Block(value1); data3 = value2; }
  ~B(){ cout << "Destructing a B " << endl;
      if (data2 == NULL) cout << "with NULL data2" << endl;
        else { cout << "with data2: " << endl; delete data2; }
      if (data3 == NULL) cout << "with NULL data3" << endl;
        else{ cout << "with data3: " << endl; data3->print(); }}
```

```
   void double_it(){data1.double_it(); data2->double_it(); data3->double_it();}
   void double_data1(){ data1.double_it(); }
   void double_data2(){ data2->double_it(); }
   void double_data3(){ data3->double_it(); } };

main(){ Block block(20); B b(10,&block);
        block.double_it(); b.double_it(); b.double_data1();
        b.double_data2(); b.double_data3(); block.double_it(); }
```

3. Δίδεται το παρακάτω ημιτελές πρόγραμμα C++:

```
#include <iostream.h>
class Investment{ double money;
public:
  Investment(double m) : money(m)
            {cout << "I can invest :" << m << " money "<< endl;}
  Investment& invest(Investment& inv)
            { cout << "Investing " << inv.money << " to " << money << endl;
              Investment* pinv = new Investment((money + inv.money)*2);
              return *pinv; }
  double money_from_investment() { return money; } };


class Person{
public:
  Person() {}
  virtual double earn() = 0; };
class Employee : public Person{ double salary;
 public :
   Employee(double money) : salary(money)
        { cout << " A Employee is created earning: " << salary << endl; }
   virtual double earn() { return salary; } };
class FreeLance : public Person{ double income; double expenses;
 public :
   FreeLance(double moneyin, double moneyout) :
        income(moneyin),expenses(moneyout)
        { cout << " A FreeLance is created getting: "
               << income << " and spending: " << expenses << endl; }
   virtual double earn() { return income-expenses;} };


class HouseHold{ double _expenses;
 public :
   HouseHold(double exp): _expenses(exp)
      { cout << " I just constructed a HouseHold spending : "
             << _expenses << endl;}
   virtual double expenses() { return _expenses;}
   virtual double investments() = 0;
   virtual double fortune() = 0; };
```

```
class Family : public HouseHold{ Person& husband; Person& wife;
                                 int NoOfKids;
  public:
   Family(int noofkids, Person& h, Person& w, double exp):
          HouseHold(exp), husband(h),wife(w),NoOfKids(noofkids)
          { cout << "I just constructed a Family With: "
                 << NoOfKids << " Kids " << endl;}
   virtual double expenses() { return HouseHold::expenses() + 10*NoOfKids;}
   virtual double investments()
          { Investment inv1(0.1*wife.earn());
            Investment inv2(0.1*husband.earn());
              return inv1.invest(inv2).money_from_investment(); }
   virtual double fortune()
          { return husband.earn()+wife.earn()+investments()-expenses();} };
class SinglePerson : public HouseHold{ Person& person;
  public:
   SinglePerson(Person& p, double exp): HouseHold(exp),person(p)
          { cout << " I just created a Single Person HouseHold " << endl;}
   virtual double investments()
          { Investment zeroinv(0);
           Investment inv(0.3*person.earn());
             return inv.invest(zeroinv).money_from_investment(); }
   virtual double fortune()
          { return person.earn()+investments()-expenses();} };

void fortune(HouseHold& nikokirio) { cout << nikokirio.fortune() << endl; }

main(){
//   Part A
//   TYPE nikokirio(PARAMS);
     fortune(nikokirio);
}
```

Τι τιμές μπορεί να πάρει το `TYPE`; Για κάθε μια από αυτές αφαιρέστε τα σχόλια από το σώμα της `main` και αντικαταστήστε με τους κατάλληλους ορισμούς το `Part A` και κατάλληλες παραμέτρους το `PARAMS` για να μπορεί να εκτελεστεί η `main`. Αιτιολογώντας την απάντησή σας, δώστε το αποτέλεσμα της εκτέλεσης.

4. Έστω ότι έχουμε να υλοποιήσουμε σε C++ μία προσομοίωση ενός ταχυφαγείου. Το ταχυφαγείο αποτελείται από ένα σύνολο από "Θέσεις εξυπηρέτησης". Οι πελάτες που εισέρχονται στο ταχυφαγείο εξυπηρετούνται ένας-ένας από τις "Θέσεις Εξυπηρέτησης" στις οποίες σχηματίζουν ουρές. Όταν εξυπηρετείται ένας πελάτης, δημιουργείται μια παραγγελία. Επίσης, ο πελάτης αφαιρείται από την ουρά της "Θέσης Εξυπηρέτησης" που περίμενε και προστίθεται σε μια λίστα πελατών του ταχυφαγείου σε αναμονή της παράδοσης της παραγγελίας του.

   Μια παραγγελία αναφέρεται στον κωδικό του μενού που περιέχει και σχετίζεται με τη "Θέση Εξυπηρέτησης" από την οποία παραγγέλθηκε. Τελικά ανατίθεται σε έναν πελάτη (οποιονδήποτε) που έχει παραγγείλει, από τη "Θέση Εξυπηρέτησης" αυτή, μενού με κωδικό ίδιο με τον κωδικό του μενού της παραγγελίας.

   Για την προσομοίωση να χρησιμοποιηθούν οι κλάσεις "Ταχυφαγείο" (`FastFoodRestaurant`), "Θέση εξυπηρέτησης" (`Counter`), "Παραγγελία" (`Order`) και "Πελάτης" (`Customer`).

Η κλάση "Ταχυφαγείο":

- έχει 5 Θέσεις εξυπηρέτησης (`counters`)
- έχει μια λίστα από πελάτες σε αναμονή της παράδοσης της παραγγελίας τους (`waiting_customers`)

Η κλάση "Ταχυφαγείο" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά η λίστα πελατών σε αναμονή είναι κενή
- Γίνεται άφιξη πελάτη, ο οποίος προστίθεται για να περιμένει στην θέση εξυπηρέτησης στην μικρότερη ουρά (`arrive`)
- Γίνεται εξυπηρέτηση, εξυπηρετώντας πελάτη από τη θέση εξυπηρέτησης με τους περισσότερους πελάτες σε αναμονή, δημιουργώντας μια νέα παραγγελία που αντιστοιχεί στην επιλογή του πελάτη αυτού, αναθέτοντάς του την παραγγελία και προσθέτοντας τον πελάτη στη λίστα πελατών εν αναμονή (`serve`)
- Παραδίδεται παραγγελία σε έναν πελάτη (οποιονδήποτε) από αυτούς εν αναμονή που ταιριάζει η παραγγελία αφαιρώντας τον από τη λίστα πελατών εν αναμονή (`deliver`)

Η κλάση "Θέση εξυπηρέτησης":

- έχει μια ουρά από πελάτες (`customer_queue`)
- έχει έναν μετρητή των πελατών που περιμένουν στην ουρά (`waiting_customers`)

Η κλάση "Θέση εξυπηρέτησης" χαρακτηρίζεται από την εξής συμπεριφορά:

- Αρχικά η ουρά των πελατών είναι κενή
- Εξυπηρετεί, αφαιρώντας πελάτη από την ουρά και μειώνοντας τον μετρητή των πελατών που περιμένουν (`serve`)
- Γίνεται άφιξη πελάτη, ο οποίος προστίθεται για να περιμένει στο τέλος της ουράς και αυξάνεται ο μετρητής των πελατών που περιμένουν (`arrive`)

Οι "Παραγγελίες" δομούνται από τον κωδικό του μενού τους και τον αριθμό της "Θέση εξυπηρέτησης" από την οποία προέκυψαν.

Οι "Πελάτες" δομούνται από τον αριθμό ταυτότητάς τους και την παραγγελία τους. Αρχικά η παραγγελία είναι κενή.

Να υλοποιήσετε σε C++ τις παραπάνω κλάσεις, ορίζοντας νέες όπου δείχνει ενδεικνυόμενο.
Σημείωση: Όπου θεωρείτε απαραίτητο, κάνετε παραδοχές στις προδιαγραφές, τεκμηριώνοντάς τις.