ΜΕΡΟΣ Β – ΓΛΩΣΣΑ C++

ΘΕΜΑ 4 (35)

Δίνεται η παρακάτω δήλωση της κλάσης Card (Card.h):

```
class Card
{
public:
   Card(); // δημιουργός
   void refund( double ); // επιστροφή ποσού στην κάρτα
   bool charge( double ); // χρέωση ποσού στην κάρτα
   double getCredit(); // συνολική οφειλή
protected:
   double credit; // η τρέχουσα οφειλή της κάρτας, default 0.0
};
```

(α) Να ορίσετε την κλάση αυτή. Η charge επιστρέφει false αν το credit ξεπερνά το 200.0 και δεν το μεταβάλλει. [10]

```
class Card
{
public:
   Card():credit(0.0){}// δημιουργός
    void refund( double a)  {
       credit-=a;
   }
       // επιστροφή ποσού στην κάρτα
    bool charge( double a)  {
       if ((credit+a)<=200)   {
       credit+=a;
       return true;
       }
       else return false;
   } // χρέωση ποσού στην κάρτα
   double getCredit() {
       return credit;
   } // συνολική οφειλή
protected:
   double credit; // η τρέχουσα οφειλή της κάρτας, default 0.0
};
```

(β) Να ορίσετε μια κλάση SmartCard ως παραγόμενη της κλάσης Card. Δεν επιτρέπεται χρέωση ή επιστροφή σε μία SmartCard, εκτός αν έχει τεθεί μια boolean μεταβλητή-μέλος approve, μέσω μεθόδου givePin(int), που να ταυτίζεται με αντίστοιχη μεταβλητή-μέλος pin της SmartCard (τίθεται στον constructor). Αν η approve είναι false, η κλήση των μεθόδων charge και refund δεν θα πρέπει να αλλάζει το credit. [15]

```
class SmartCard : public Card
{
public:
    SmartCard (int a):pin(a) {}
    void givePin (int a) {
        if (a==pin) {
            approve = true;
        }
    }
   void refund( double a) {
```

```
        if (approve) credit-=a;
    }
        // επιστροφή ποσού στην κάρτα
    bool charge( double a) {
        if (approve && (credit+a)<=100) {
        credit+=a;
        return true;
        }
        else return false;
    } // χρέωση ποσού στην κάρτα

private:
     int pin;
     bool approve = false;
};
```

(γ) Θα μπορούσε η μεταβλητή credit να είναι private; Αιτιολογήστε. [5]

Δεν θα μπορούσε, γιατί δεν θα επιτρεπόταν η απευθείας πρόσβαση σε αυτήν από την υποκλάση SmartCard, εκτός αν οι μέθοδοι charge και refund της SmartCard χρησιμοποιούν τις αντίστοιχες μεθόδους της υπερκλάσης.

(δ) Υπερφορτώστε τον τελεστή + ως μέλος της κλάσης, ώστε συμπεριφέρεται όπως η charge. [5]

```
bool operator+(double a) {
               return charge(a); }
```

ΘΕΜΑ 5 (30)

Δίνεται η παρακάτω main (θεωρείται ότι γίνονται include οι δηλώσεις των κλάσεων του προηγούμενου θέματος).

(α) Τι θα τυπωθεί σε κάθε μια από τις περιπτώσεις Α και Β; [5]

A.

```
#include <iostream>
using namespace std;
int main()
{
   Card card1;
   Card * card2 = new
         SmartCard(1234);
   card1.charge(210.0); //-10
   card1.refund (10.0);
   cout <<
   card1.getCredit()<<endl;
   card2->charge(110.0);
   card2->refund(100.0);
 cout << card2->getCredit();
//10
 delete card2;
}
```

B.

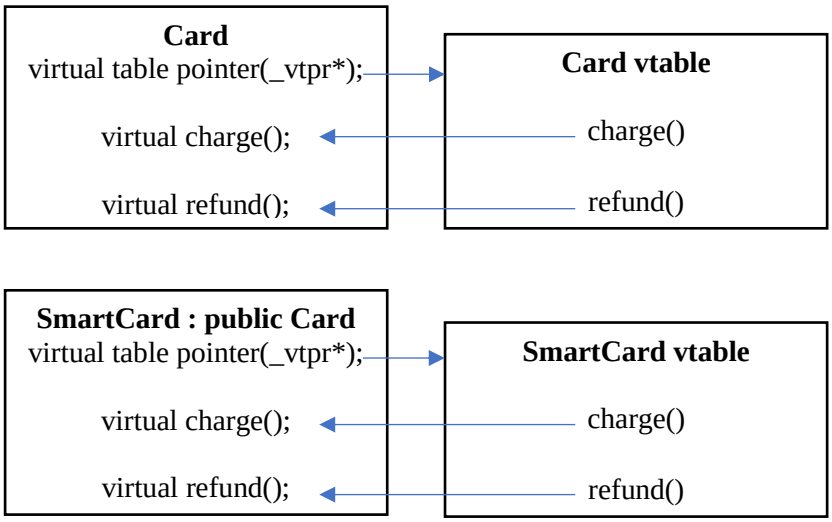
```
#include <iostream>
using namespace std;
int main()
{
   Card card1;
   SmartCard card2(1234);

   card1.charge(190.0);
   card1.charge(10.0);
   cout <<
   card1.getCredit()<<endl;
//200
   card2.charge(110.0);
   card2.givePin(1234);
   card2.refund(100.0);
 cout << card2.getCredit();
//-100
}
```

(β) Τι θα τυπωθεί σε κάθε μια από τις περιπτώσεις Α και Β αν οι μέθοδοι charge και refund δηλωθούν ως εικονικές (virtual, όπως επίσης και ο destructor); [10]

A.	**B.**
`//-10` `//0`	`//200` `//-100`

(γ) Να σχεδιάσετε τον πίνακα εικονικών μεθόδων (vtable) για τις κλάσεις Card και SmartCard. [5]



(δ) Να καταγράψετε τη σειρά κλήσης των constructors και των destructors για τα δύο αντικείμενα σε κάθε μια από τις περιπτώσεις Α και Β. [10]

A.	**B.**

`Card()`
`Card(), SmartCard()`
`~SmartCard(), ~Card() //delete`
`~Card() //out of scope`
	`Card()`
	`Card(), SmartCard()`
	`~SmartCard(), ~Card() //out of scope`
	`~Card() //out of scope, created first`

ΘΕΜΑ 6 (35)

Δίνεται η δήλωση της παρακάτω πρότυπης κλάσης (template class) Order.

```
#include <vector>
using namespace std;
template <class T>
class Order
{
public:
  Order();
  void reorder();
  ~ Order();
private:
   vector<T *> elements;
};
```

(α) Να ορίσετε constructor και destructor. [10]

```
template <class T>
Order<T>::Order() {

    elements.push_back(new
string("a1"));
    elements.push_back(new
string("a2"));
    elements.push_back(new
string("a3"));
    elements.push_back(new
string("a4"));
    elements.push_back(new
string("a5"));
}
```

```
template <class T>
Order<T>::~Order() {
    for (int i=0; i++; i<5) {
        delete elements[i];
    }
}
```

(β) Υλοποιήστε τη μέθοδο reorder, ώστε να αντιστρέφει τη σειρά στα N/2 τελευταία στοιχεία. [15]

```
template <class T>
void Order<T>::reorder() {
    vector<T*> temp (elements.begin(), elements.begin() +
elements.size()/2);

    for (int i=elements.size()-1; i>elements.size()/2-1; i--)
    {
        temp.push_back(elements[i]);
    }
    elements = temp;
}
```

(γ) Γράψτε main που να δημιουργεί ένα αντικείμενο order με 5 strings a1, a2,…, a5 και να καλεί την reorder. [10]

```
int main () {
    Order<string> o;
    o.reorder();
}
```

Μπορείτε να κάνετε χρήση των μεθόδων at(), push_back(), pop_back (), καθώς και iterator, begin(), end().Να χρησιμοποιήσετε δυναμική διαχείριση μνήμης και να αποφύγετε τυχόν διαρροές.