Задача 2. Задачата да се реши на езика C++.

А) Разгледайте дадения по-долу фрагмент. Оградете всички твърдения за него, които са верни. Всяко вярно тяърдение, което е маркирано, увеличава резултата за задачата. Всяко грешно твърдение, което е маркирано, го намалява.

```
#include <string>

// Представя предмет
class Item {
    // Етикет на предмета
    std::string label;

    virtual int hash() = 0;
};

// Представя молив
class Pencil : protected Item {
};

// Представя кутия
class Box : public Item {
    // Сочи към това, което
    // се пази в кутията
    Item *contents;
public:
    // Съхранява something в кутията
    void store(Item *something) {
        contents = something;
    }
};
```

А) Член-променливата label на класа Item е protected.

Б) Един обект от тип Pencil може да се съхрани в себе си.

В) Един обект от тип Box може да се съхрани сам в себе си.

Г) Един обект от тип Pencil може да се съхрани в две кутии едновременно.

Д) От класа Box НЕ МОГАТ да се създават обекти, но могат да се създа- ват указатели (pointer) и псевдони- ми (reference).

Е) Кодът за Pencil ще предизвика компилационна грешка, защото в тяло- то на класа не е дефинирано нищо.

Ж) Един обект от тип Box може да се съхрани в друг обект от тип Box.

З) Класът Pencil наследява виртуално класа Item.

И) Всеки обект от тип Pencil има член label, който е protected.

К) Класът Pencil е абстрактен.

Л) Обекти от тип Item не могат да се създават, защото в класа не е дефи- ниран нито един конструктор.

М) ПрисъздаванетонаобектоттипBox не се задава стойност за contents и тази променлива остава неинициа- лизирана.

Б) Разгледайте дадения по-долу фрагмент. Някои от редовете на main() са номерирани в коментар. За всеки от тях посочете какво ще се изведе на стандартния изход след неговото изпълнение.

```
#include <iostream>

class base {
public:
    base() { std::cout << "base::base()\n"; }

    ~base() { std::cout << "base::~base()\n"; }
```

```
    virtual void f() { std::cout << "base::f()\n"; };
};

class derived : public base {
public:
    derived() { std::cout << "derived::derived()\n"; }

    ~derived() { std::cout << "derived::~derived()\n"; }

    virtual void f() { std::cout << "derived::f()\n"; }
};

int main() {
    base *p = new derived(); //1
    p->f();   //2
    base obj = *p;
    obj.f();  //3
    delete p; //4
}
```

Ред //1 ще изведе:

Ред //2 ще изведе:

Ред //3 ще изведе:

Ред //4 ще изведе:

Решение.

А)

А) Грешно. Член-променливата label е private, тъй като се намира в default scope-а на class, който е private.

Б) Грешно. Един обект от тип Pencil не може да се съхрани в себе си, тъй като не съхранява указател към обект от тип Item или Pencil.

В) Вярно. Един обект от тип Box може да се съхрани в себе си, тъй като съхранява указател към обект от тип Item, който може да бъде обект от тип Box.

Г) Вярно. Възможно е един обект от тип Pencil да се съдържа в два различни обекта от тип Box, всеки един от тях държейки указател към обекта от тип Pencil, който е директен наследник на класа Item.

Д) Вярно. От класа Box НЕ МОГАТ да се създават обекти, защото той е абстрактен (не е предефинирал виртуалната функция в Item), но могат да се създават указатели и псевдоними.

Е) Грешно. В тялото на класа Pencil може да е празно (не е задължително да се предефинира виртуалната функция на базовия клас Item). Няма да възникне компилационна грешка.

Ж) Вярно. Един обект от тип Box съхранява указател към обект от тип Item, който може да бъде обект от тип Box. Следователно един Box обект е напълно възможно да се съхрани в друг от същия тип.

З) Грешно. Класът Pencil наследява класа Item с protected спецификатор.

И) Грешно. Всеки обект от тип Pencil наследява private член-променлива label от класа Item.

К) Вярно. Класът Pencil е абстрактен, тъй като не предефинира виртуалната функция на базовия си клас.

Л) Грешно. Обекти от тип Item не могат да се създават, тъй като са абстрактни.

М) Грешно. Обекти от тип Box не могат да се създават. Ако можеха, това твърдение щеше да е вярно.

Б)

Ред //1 ще изведе:
base::base()
derived::derived()

Коментар: Първо се извиква конструктура на базовия клас и след това на наследника.

Ред //2 ще изведе:
derived::f()

Коментар: Няма да се извика f() на базовия клас, тъй като p държи обект от тип derived, който е пренаписал f().

Ред //3 ще изведе:
base::f()

Коментар: Обектът от тип derived ще се преобразува до base в горния ред и ще се извика функцията f() на класа base.

Ред //4 ще изведе:
base::~base()
base::~base()

Коментар: Веднъж ще се извика деструктора на базовия клас заради указателя към базовия клас и още веднъж за обекта obj, който е от тип base.

□