Задача 2. Задачата да се реши на езика C++. Отговорите да се попълнят в посочените за тях полета. Оценява се само попълненото в полетата. Всичко друго писано по листа не носи точки.

2A) Разгледайте програмата. Някои от конструкциите във функцията main са номерирани в коментар. За всеки такъв ред посочете какво ще се изведе в резултат от неговото изпълнение. Отговор, който не съответства точно на това, което извежда програмата, носи нула точки. Ако смятате, че някой от редовете няма да изведе нищо, напишете „нищо“. Ако смятате, че някой от тях ще предизвика грешка, опишете каква. Ако за някоя конструкция не бъде попълнено нищо, отговорът не носи точки.

```
#include <iostream>

using std::cout;

class test {
public:
    int var;

    test() {
        cout << "test()\n";
        var = 0;
    }

    test &operator=(const test &other) {
        if (this != &other) {
            cout << "copy\n";
            var = other.var;
        } else {
            cout << "self-assignment\n";
        }
        return *this;
    }

    static test &instance() {
        static test obj;
        return obj;
    }

    test &self() {
        return *this;
    }

    void _oncreate() {
        cout << "_oncreate()\n";
    }

    void _oncopy() {
        cout << "_oncopy()\n";
    }
};

int main() {
    test &r1 = test::instance(); // 1
    test &r2 = test::instance(); // 2
    r1.var = 10; // 3
    cout << r2.var << "\n"; // 4
    r1.self() = r2; // 5
    new test[3]; // 6
}
```

Ред // 1 ще изведе: **test()**

Ред // 2 ще изведе: **нищо**
(обекта е статичен и няма да се създаде нов, т.е. няма да се извика конструктура му)

Ред // 3 ще изведе: **нищо**
(присвоява се стойност на променлива, която не предизвиква странична реакция)

Ред // 4 ще изведе: **10**
(r1 и r2 сочат към една и съща променлива (структура) в паметта)

Ред // 5 ще изведе: **self-assignment**
(по същата причина както в ред 4)

Ред // 5 ще изведе:
test()
test()
test()
(Извиква се конструктура по подразбиране за всеки елемент от масива new test[3], към който не се пази указател, но това не означава, че не се създава)

2Б) Възможно ли е в един клас да се дефинират няколко различни копиращи конструктура?
Отговор: а) Да.

```
#include <iostream>
#include <string>
#include <utility>

using namespace std;

class Car {
   string brand;

public:
   explicit Car(string brand) {
      this->brand = move(brand);
   }

   void setBrand(string newBrand) {
      this->brand = std::move(newBrand);
   }

   string getBrand() {
      return brand;
   }

   Car(const Car &car) { // First copy constructor
      brand = car.brand;
   }

   Car(Car &car) { // Second copy constructor
      brand = car.brand + " limited";
   }

   friend ostream &operator<<(ostream &os, const Car &car) {
      os << car.brand << endl;
      return os;
   }
};

int main() {
   const Car car1("audi");
   Car car2("porsche");

   Car newCar1(car1);
   Car newCar2(car2);

   cout << newCar1 << newCar2;
}
```

2В) Нека са дадени следните дефиниции:

```
class foo {
public:
   foo(int) {}
};

void g(foo) {}
void f(int) {}
```

Срещу всеки от изразите посочете (като напишете „да“ или „не“) дали ще се компилира:

g(5); **ДА** (g приема клас foo, който се инициализира с int)
f(foo(5)); **НЕ** (foo(5) е клас, т.е. не връща int, който се да приеме функцията f)
foo('a'); **ДА** ('a' е числото 97 в ASCII таблицата)

□