Задача 1 (Вариант А) (20 точки)

С 1-2 изречения отговорете на следният въпрос:
Спазен ли е принципа **S** от **SOLID** принципите (напишете кой е той), в следния код ? Ако не е спазен, опишете защо.

```
class ItemHolder{
public:
    void print_and_clear_vector();
private:
    std::vector<int*> items;

};
void ItemHolder::print_and_clear_vector()
{
    for(int* item: this->items)
    {
        std::cout << *item << " ";
    }
    for(int* item: this->items)
    {
        delete item;
    }
}
```

Задача 2 (Вариант А) (20 точки)

Какво ще изведе следната програма на екрана ? Защо ?

```
#include <iostream>
#include <vector>

class Bar{
public:
    int value;
};

template <class T>
void print(const T& to_print)
{
    std::cout << to_print << std::endl;
}

template <>
void print(const int& to_print)
{
    std::cout << to_print + 10 << std::endl;
}

template <>
void print(const double& to_print)
{
```

```
    std::cout << to_print + 3.14 << std::endl;
}


template <>
void print(const Bar& to_print)
{
    std::cout << to_print.value + 3.14 << std::endl;
}

int main()
{
    Bar bar;
    bar.value = -3;

    print("Hello !");
    print(bar);
    print(bar.value);
}
```

Задача 1 (Вариант Б) (20 точки)

С 1-2 изречения отговорете на следният въпрос:
Спазен ли е принципа **O** от **SOLID** принципите (напишете кой е той), в следния код ? Ако не е спазен, опишете защо.

```
#include <iostream>

class Person {
public:
    std::string name;
    virtual ~Person() { };
};

class Student : public Person {
public:
    int id;
};

class Professor : public Person {
public:
    std::string title;
};

class Serializer {
public:
    static std::string toString(Person* person) {
        std::string output;

        Student* student_ptr = dynamic_cast<Student*>(person);
        Professor* prof_ptr = dynamic_cast<Professor*>(person);
        if (student_ptr) {
            output = "student:" + student_ptr->name + ":id:" +
std::to_string(student_ptr->id);
```

```
        } else if (prof_ptr) {
            output = "professor:" +prof_ptr->name + "title:" + prof_ptr->title;
        } else {
            output = "person " + person->name;
        }

        return output;
    }
};

int main() {
    Person person;
    person.name = "Bob";

    Student student;
    student.name = "Maria";
    student.id = 77777;

    std::cout << Serializer::toString(&person) << "\n";
    std::cout << Serializer::toString(&student) << "\n";
}
```

Задача 2 (Вариант Б) (20 точки)

Какво прави следната програма?

```
#include <iostream>
#include <fstream>
#include <string>
template <class T>
class Printer {
   private:
        std::ostream* outStream;
public:
    Printer(std::ostream& outputStream) {
        this->outStream = &outputStream;
    }

    void print(T data) {
        *(this->outStream) << data;
    }
};

int main() {
    Printer<int> printer1(std::cout);
    printer1.print(123);

    std::ofstream outStream("f.out");
    Printer<std::string> printer2(outStream);
    printer2.print("This is a very ");
    printer2.print("important sentence");

    Printer<std::string> printer3(std::cout);
    printer3.print("you've got mail");

    outStream.close();
```

```
    printer2.print("This is another sentence");
}
```

Задача 3 (80 точки)

Не е позволено използването на STL. Позволено е използването на библиотеката , при необходимост

Напишете клас `ProtectedValue`, който съдържда като член данни:

- Стойност (value), чийто тип е избран от потребителя
- Код за сигурност (code), който е цяло положително число, зададено от потребителя

Достъпът до `value` на `ProtectedValue` се случва само през метод, в който е подаден код от потребителя. Този код трябва да се сравни с `code` на текущата инстанция. Ако двата кода са равни, само тогава трябва да се върне стойността. В противен случай, се връща стойност по подразбиране или се хвърля `std::invalid_argument` (И двата подхода ще се приемат за верни).

Задаването на `value` се случва само при създаването на обекта, като тогава също се подава и кода за сигурност. ()

Напишете клас `ProtectedArray` - разширяващ се масив от `ProtectedValue`. Потребителят може да добавя нови елементи към масива чрез стойност и код, като се спазват същите правила за създаване, както при `ProtectedValue`. Потребителят може да достъпва елементите на `ProtectedArray` чрез метод, който приема два аргумента:

- Индекс, който да бъде достъпен в масива
- Код за сигурност

Ако кодът за сигурност съвпадне, да се върне стойността. Ако кодът за сигурност не съвпадне, да се покаже подходящо съобщение и да се върне стойност по подразбиране.

Задача 4 (80 точки)

Позволено е използването на STL

Напишете програма за управление на автосервиз.

В нашия автосервиз могат да бъдат обслужвани автомобили и микробуси. При влизането на всяко превозно средство в сервиза, за него бива въведена следната информация:

- Марка на превозното средство
- Модел на превозното средство
- Година на производство
- Проблем - проблемът е описан в низ, като свободен текст
- Сериозност на проблема - Нисък, Среден, Висок

За автомобилът се въвежда и допълнителна информация:

- Дали автомобилът е на частно лице или на фирма

За микробуса се въвежда и допълнителна информация:

- Дали микробусът превозва пътници или не

Напишете клас, който да пази необходимата информация на сервиз. Направете методи за въвеждане на ново превозно средство, което влиза в сервиза. Направете метод, който показва дали сервиза има капацитет да поеме нов ремонт. Капацитета се изчислява по следния начин, спрямо сериозността на проблема:

- Ниска сериозност на проблема носи 1 точка
- Средна сериозност на проблема носи 3 точки
- Висока сериозност на проблема носи 5 точки

Сервиза може да има най-много 10 точки в даден момент.

Бонус: Направете класа за сервиза да е Singleton