Задача 1 (20 точки)

С едно-две изречения отговорете на следния въпрос: Какво ще изведе следния код ? Защо ?

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

class Foo
{
public:
    Foo()
    {
        std::cout << "Foo called !\n";
    }
};

class Bar{
public:
    Bar()
    {
        std::cout << "Bar called !\n";
    }
};

class Baz: Bar{
public:
    Foo* temp;
    Baz()
    {
        temp = new Foo[3];
    }
};
int main()
{
    Baz temp;
    return 0;
}
```

Задача 2 (20 точки)

С едно-две изречения отговорете на следния въпрос: Какви са грешките в следният код ? Обяснете защо е грешен кода:

```
class Foo
{
public:
    Foo()
    {
        this->data = new int(5);
    }

    ~Foo()
```

```
    {
        delete data;
    }

protected:
    int* data;
};

class Bar: public Foo{
public:
    Bar(): Foo()
    {
        this->more_data = new int(-3);
    }

    ~Bar()
    {
        delete this->more_data;
    }
private:
    int* more_data;
};

int main()
{
    std::vector<Foo*> container;

    Foo* temp = new Bar;
    container.push_back(temp);

    delete temp;
    return 0;
}
```

Задача 3 (80 точки)

Не е позволено използването на STL.

Ще дефинираме следните аритметични операции върху неизвестен брой числа:

- събиране (събира всички числа) - върху числата `1, 2 и 4`, операцията събиране би извела `7`
- изваждане (изважда всички числа) - върху числата `1, 2 и 4`, операцията изваждане би извела `-7`
- умножение (умножава всички числа) - върху числата `1, 2 и 4`, операцията умножение би извела `8`

Напишете клас `IntegerOperation`, който извършва аритметични операции върху неизвестен брой цели числа, които се пазят в класа. Потребителят може да добавя числа, върху които да бъдат извършвани различни операции. Резултатът от операциите също се пази в класа. Потребителят може да взима резултата от текущата инстация.

Пренапишете следните оператори за класа `IntegerOperation`:

- оператор+ (извършва операция събиране върху двата обекта, и след това връща сбора от резултатите им)
- оператор- (извършва операция изваждане върху двата обекта, и след това връща разликата от резултатите им)
- оператор* (извършва операция умножение върху двата обекта, и след това връща произведението от резултатите им)

Тестов сценарий 1:

```
    IntegerOperation test;
    test.insert(5);
    test.insert(3);
    test.insert(7);

    test.sum();
    std::cout << test.get_result() << std::endl;

    test.sub();
    std::cout << test.get_result() << std::endl;

    test.multiply();
    std::cout << test.get_result() << std::endl;
```

Изход:

```
15
-15
105
```

Тестов сценарий 2:

```
    IntegerOperation test;
    test.insert(5);
    test.insert(3);
    test.insert(7);


    IntegerOperation test2;
    test2.insert(2);
    test2.insert(14);
    test2.insert(4);

    std::cout << test + test2 << std::endl;
    std::cout << test - test2 << std::endl;
    std::cout << test * test2 << std::endl;
```

Изход:

```
35
5
11760
```

Задача 4 (80 точки)

Позволено е използването на STL

Дадено е да моделирате игра с карти (в стил Magic the gathering, Heartstone, etc.).

Всяка карта има следните свойства:

- Име
- Номер на картата (цяло положително число)
- Номер на изображението на картата (цяло положително число)
- При поискване, на екрана се отпечатва цялата необходима информация за картата

Имаме два вида карти - карти с герои, и карти с магии.

Картата с герой има следните свойства:

- Атакуваща сила (цяло положително число)
- Защитна сила (цяло положително число)

Картата с магия има следните свойства:

- Описание на ефекта (низ)

Освен това, имаме и трети вид карти - специални.

Специалните карти са едновременно карта с герой и карта с магия. Специалните карти имат и ниво (цяло положително число)

Потребителят има и тесте от карти. Потребителят може да добавя карта към тестето си. При поискване, на екрана се отпечатва цялата информация за всяка една от картите в тестето.

Бонус: при реализирането на класа за тестето, използвайте Singleton