Задача1(Бензоколонка)

Напишите класс, описывающий бензоколонку. Прототип класса:

```
class GasStation {
public:
    // Конструктор, принимающий один параметр - ёмкость резервуара колонки
    // Резервуар создаётся пустой

    // Залить в резервуар колонки n литров топлива
    // Если столько не влезает в резервуар - не заливать ничего, выбросить std::exception
    void fill(unsigned int n);

    // Заправиться, забрав при этом из резервура n литров топлива
    // Если столько нет в резервуаре - не забирать из резервуара ничего, выбросить
std::exception
    void tank(unsigned int n);

    // Запросить остаток топлива в резервуаре
    unsigned int get_limit() const;
};
```

Вариант кода для локального тестирования:

```
GasStation s = GasStation(1000);            // Колонка на 1000, пока пустая
s.fill(300);                                // Это влезет в резервуар,
cout << s.get_limit() << endl;              // ... так что здесь увидим на экране 300.

s.tank(100);                                // Забрали из резервуара 100,
s.fill(300);                                // ... после чего долили ещё 300,
cout << s.get_limit() << endl;              // ... так что на экране ожидаем 500.

for(unsigned int i = 0; i < 5; i++)         // В цикле забрали 5 раз по 50,
    s.tank(50);
cout << s.get_limit() << endl;              // ... так что на экране ожидаем 250.

s.fill(1000);                               // А вот тут ожидаем exception.
                                            // (После проверки exception-а строку стоит просто
убрать.)

for(unsigned int i = 0; i < 50; i++) {      // Теперь пытаемся забрать 50 раз по 100,
    s.tank(100);                            // из-за чего на третьей попытке ждём exception.
    cout << s.get_limit() << endl;
}
```

Например:

Ввод	Результат
1000 6 fill 2000 fill 500 tank 200 tank 200 tank 200 tank 100	Creating gas station, capacity: 1000 Trying to fill 2000 Failed to fill Fuel limit after the action is 0 Trying to fill 500 Fuel limit after the action is 500 Trying to tank 200 Fuel limit after the action is 300 Trying to tank 200 Fuel limit after the action is 100 Trying to tank 200 Failed to tank Fuel limit after the action is 100 Trying to tank 100 Fuel limit after the action is 0

Задача 2(Копилка)

Реализуйте класс, описывающий копилку. Прототип класса:

```
class MoneyBox {
public:
    // Конструктор и деструктор, если нужны

    // Добавить монетку достоинством value
    void addCoin(unsigned int value);

    // Получить текущее количество монеток в копилке
    unsigned int getCoinsNumber() const;

    // Получить текущее общее достоинство всех монеток
    unsigned int getCoinsValue() const;
};
```

Для тестирования можете использовать следующий базовый пример:

```
    MoneyBox m;
    // Добавили монетку достоинством 10
    m.addCoin(10);
    // Добавили монетку достоинством 5
    m.addCoin(5);

    // Ожидаем, что монеток внутри 2 штуки
    cout << m.getCoinsNumber() << endl;
    // Ожидаем, что общее достоинство всех монеток 15
    cout << m.getCoinsValue() << endl;
```

Данный пример должен вывести:

```
2
15
```

Например:

Ввод	Результат
3 1 10 20	Adding coin with value 1 Adding coin with value 10 Adding coin with value 20 Number of coins: 3 Value of coins: 31

Задача 3(Сфера)

Задан следующий класс точки в трёхмерном пространстве:

```
class Point
{
protected:
    double _x;
    double _y;
    double _z;

public:
    Point(double x, double y, double z): _x(x), _y(y), _z(z) {
    }

    double x() const {
        return _x;
    }

    double y() const {
        return _y;
    }

    double z() const {
        return _z;
    }
};
```

Реализуйте класс, задающий сферу. Прототип класса:

```
class Sphere
{
public:
    // Конструктор сферы с центром в точке center и радиусом r
    Sphere(const Point& center, double r);

    // Проверка, попадает ли заданная точка p в данную сферу.
    // (Расстояния сравнивать с точностью 1e-6.)
    bool covers(const Point& p) const;
};
```

Для тестирования можете использовать следующий базовый пример:

```
    // Создаём сферу
    Point center(10.0, 10.0, 10.0);
    Sphere s(center, 0.5);

    // Создаём пару тестовых точек
    Point p1(10.1, 10.1, 10.1);
    Point p2(2, 2, 2);

    // Ожидаем, что первая точка внутри сферы
    cout << boolalpha << s.covers(p1) << endl;
    // Ожидаем, что вторая точка не попала внутрь сферы
    cout << boolalpha << s.covers(p2) << endl;
```

Данный пример должен вывести:

```
true
false
```

Например:

Ввод	Результат
10.0 10.0 10.0 0.5 2 10.1 10.1 10.1 2.0 2.0 2.0	Creating sphere with center (10.00; 10.00; 10.00) and R = 0.50 Testing point (10.10; 10.10; 10.10). The point is inside sphere: true Testing point (2.00; 2.00; 2.00). The point is