**Разработка фреймворка юнит-тестов**

**ОБЩИЙ БАЛЛ 13**

1.

Вопрос 1

В чём недостатки приведённой ниже реализации шаблона AssertEqual?

template <typename T>

void AssertEqual(const T& first, const T& second) {

if (first != second) {

ostringstream os;

os << "Assertion failed: " << first << " != " << second;

throw runtime\_error(os.str());

}

}

4 балла



Аргументы принимаются по значению, хотя в этом нет необходимости



Аргументы first и second обязаны быть одного типа



Эта реализация **не** помогает понять, какой именно assert сработал



Эта реализация не сообщает значения аргументов, если они оказались не равны

2.

Вопрос 2

Почему функцию AssertEqual мы сделали шаблонной, а функцию Assert — нет?

3 балла



AssertEqual должна уметь сравнивать аргументы любых типов, а Assert проверяет истинность аргументов только одного типа — типа bool



Функция Assert реализована через AssertEqual, поэтому она тоже является шаблонной



Функции Assert не надо проверять свои аргументы на равенство

3.

Вопрос 3

Как нам удалось добиться запуска всех юнит-тестов даже тогда, когда некоторые из них падают?

2 балла



Мы обернули вызов каждого теста в блок try/catch, который ловит исключение, выбрасываемое из функций Assert и AssertEqual



Мы не прерываем выполнение теста, если сработал assert



Мы стали передавать тесты в шаблонную функцию RunTest

4.

Вопрос 4

С какой целью мы обернули блок try/catch в метод RunTest и почему сделали его шаблонным?

4 балла



Чтобы избежать дублирования кода при оборачивании вызова каждого юнит-теста в блок try/catch



Чтобы падение одних юнит-тестов не мешало запуску других



Чтобы передавать юнит-тесты в качестве параметров, не указывая явно их тип



Чтобы выводить более понятные сообщения в случае падения тестов