

```

public class operation_test{
    [Fact] Iub [Theory][InlineData(var1,var2)]
    public void nazwa_testu() {
        //arrange
        _fixture.arrange_operation();

        act();

        //assert
        _fixture.assert();
    }

    private void act(){
        _fixture.act();
    }

    public operation_test(){
        _fixture = new operation_test_fixture();
    }

    private readonly operation_test_fixture _fixture;
}

```

```

public class operation_test_fixture{
    private Operation _operation;

    public operation_test_fixture(){
        _operation = new Operation();
    }

    public void arrange_operation() {
        _operation.Set(0);
    }

    public void assert() {
        _operation.Result.Should().Be(0);
    }

    private void act(){
        _operation.Run();
    }
}

```

```

public class operation_test{
    [Fact] Iub [Theory][InlineData(var1,var2)]
    public void nazwa_testu()    {
        //arrange
        _fixture.arrange_operation();

        act();

        //assert
        _fixture.assert_throw_exception();
    }

    private void act() {
        _fixture.act();
    }

    public OperationText(){
        _fixture = new operation_test_fixture();
    }

    private readonly operation_test_fixture _fixture;
}

```

```

public class operation_test_fixture{
    private Operation _operation;
    private Action _act;

    public operation_test_fixture() {
        _operation = new Operation();
    }

    public void arrange_operation() {
        _operation.Set(0);
    }

    public void assert_throw_exception() {
        _act.ShouldThrow<Exception>().WithMessage("xyz");
    }

    private void act() {
        _act = () => operation.Set(0);
    }
}

```

nazwa_testu

fakt_jaki_testuje

co_testuje_i_jak

poprawny_wynik_dodawania_gdy_liczba_a_i_liczba_b_są_całkowite

nazwa_asercji

asercja

co_weryfikuje

asercja_poprawny_wynik_dodawania