CSE 230 Problem Set 09

Problem 25.1: Compute Pay

Consider the following C++:

```
double computePay(double hours, double wage)
{
   if (hours < 40.0)
      return hours * wage;
   else
      return (wage * 40.0) + (wage * 1.5 * (hours – 40.0));
}
```

Create unit tests to exercise the following test cases. Put all the unit tests in a single function.

NAME	INPUT (HOUR, WAGE)	OUTPUT
Zeros	0, $0.00	$0.00
No time	0, $8.00	$0.00
One hour	1, $8.00	$8.00
No wage	1, $0.00	$0.00
Just under full time	39, $10.00	$390.00
Full time	40, $10.00	$400.00
One hour overtime	41, $10.00	$415.00
Double time	80, $10.00	$1,000.00
Negative hours	-1, $10.00	error
Negative wage	1, -$8.00	error
Unreasonable hours	168, $10.00	error

```
void test_computePay()
{
    assert (computePay(0,0.00) == 0.00);
    assert (computePay(0, 8.00) == 0.00);
    assert (computePay(1, 8.00) == 8.00);
    assert (computePay(1,0.00) == 0.00);
    assert (computePay(39, 10.00) == 390.00);
    assert (comptuePay(40,10.00) == 400.00);
    assert (computePay(41, 10.00) == 415.00);
    assert (computePay(80,10.00) == 1000.00);
    assert (computePay(-1,10.00) == null);
    assert(computePay(1,-8.00) == null);
    assert (computePay(168, 10.00) == null);)
}
```

Problem 25.2: Percent

Consider the following class:

```
class Percent
{
   public:
      double percent;              // stored in value 0.0 – 1.0
      Percent() : percent(0.0) {}
      double get() { return percent * 100.0;}
      void set(double input)
      {
         if (input >= 0.0 && input <= 100.0)
            percent = input / 100.0;
      }
};
```

Identify test cases for all the methods in the class.

NAME	SETUP	EXERCISE	VERIFY
Get Default	myPercent = Percent()	returnValue = myPercent.get()	returnValue = 0.0 myPercent.percent = 0.0
Get 50%	myPercent = Percent() myPercent.percent = 0.5	returnValue = myPercent.get()	returnValue = 50.0 myPercent.percent = 0.5
Get 100%	myPercent = Percent() myPercent.percent = 1.0	returnValue = myPercent.get()	returnValue = 100.0 myPercent.percent = 1.0
Set 50%	myPercent = Percent()	myPercent.set(50.0)	myPercent.percent = 0.5
Set 100%	myPercent = Percent()	myPercent.set(100.0)	myPercent.percent = 1
Higher than 100%	myPercent = Percent()	myPercent.set(101.0	Error, myPercent.percent = 0.0
Lower than 0%	myPercent = Percent()	myPercent.set(-1.0)	Error, myPercent.percent = 0.0

Show two unit tests for the `set()` method. Make sure that each has the four parts (setup, exercise, verify, teardown). Each unit test should be in its own method in a `TestPercent` class.

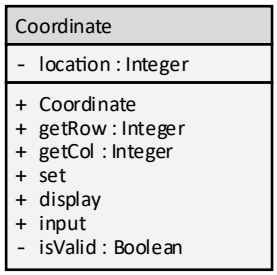
```
void TestPercent::test_getter()
{
//Get default

        //Setup
        Percent myPercent = Percent();

        //Exercise
        double returnValue = myPercent.get();

        //Verify
        assert(returnValue == 0.0);
        assert(myPercent.percent == 0.0);

    //Get 50%

        //Setup
        Percent myPercent = Percent();

        //Exercise
        double returnValue = myPercent.get();

        //Verify
        assert(returnValue == 0.0);
```

```
        assert(myPercent.percent == 0.0);

    //Get 100%

        //Setup
        Percent myPercent = Percent();
        myPercent.percent = 1.0;

        //Exercise
        double returnValue = myPercent.get();

        //Verify
        assert(returnValue == 100.0);
        assert(myPercent.percent == 1.0);}
```

```
void TestPercent::test_setter()
{
//Set 50%

        //Setup
        Percent myPercent = Percent();

        //Exercise
        myPercent.set(50.0);

        //Verify
        assert(myPercent.percent == 0.5);

    //Set 100%

        //Setup
        Percent myPercent = Percent();

        //Exercise
        myPercent.set(100.0);

        //Verify
        assert(myPercent.percent == 1.0);

//Higher than 100%

        //Setup
        Percent myPercent = Percent();

        //Exercise
        myPercent.set(101.0);

        //Verify
        assert(myPercent.percent == 0.0);

//Lower than 0%

        //Setup
        Percent myPercent = Percent();

        //Exercise
        myPercent.set(-1.0);

        //Verify
        assert(myPercent.percent == 0.0);

}
```

Problem 25.3: Coordinate

Consider the following class diagram designed to represent the position of a piece on a chess board:



Enumerate a set of test cases for each of the public methods:

METHOD UNDER TEST	TEST NAME
Coordinate	test_constructor (initialize location as 0)
get row	test_get_row_zero
	test_get_row_one
	test_get_row_two
	test_get_row_three
	test_get_row_four
	test_get_row_five
	test_get_row_six
	test_get_row_seven
get col	test_get_col_zero
	test_get_col_one
	test_get_col_two
	test_get_col_three
	test_get_col_four
	test_get_col_five
	test_get_col_six
	test_get_col_seven
set	test_set_valid_coordinates
	test_set_invalid_row
	test_set_invalid_col
	test_set_negative_coordinates
	test_set_too_large_coordinates
display	test_display (ensure location wasn't affected by display)

Create a test runner as was done in example 25.3. The class name will be `TestCoordinate`.

```
Class TestCoordinate: public TestCase {
Public:
    Void run()
    {
        test_get_row_zero();
        test_get_row_one();
        test_get_row_twp();
        test_get_row_three();
        test_get_row_four();
        test_get_row_five();
        test_get_row_six();
        test_get_row_seven();

        test_get_column_zero();
        test_get_column_one();
        test_get_column_twp();
        test_get_column_three();
        test_get_column_four();
        test_get_column_five();
        test_get_column_six();
        test_get_column_seven();

        test_set_valid_coordinates();
        test_set_invalid_row();
        test_set_invalid_col();
        test_set_negative_coordiantes();
        test_set_too_large_coordinates();
        test_display();

    }
private:
    void test_get_row_zero();
    void test_get_row_one();
    void test_get_row_twp();
    void test_get_row_three();
    void test_get_row_four();
    void test_get_row_five();
    void test_get_row_six();
    void test_get_row_seven();

    void test_get_column_zero();
    void test_get_column_one();
    void test_get_column_twp();
    void test_get_column_three();
    void test_get_column_four();
    void test_get_column_five();
    void test_get_column_six();
    void test_get_column_seven();

    void test_set_valid_coordinates();
    void test_set_invalid_row();
    void test_set_invalid_col();
    void test_set_negative_coordiantes();
    void test_set_too_large_coordinates();

    void test_display();
}
```