Functions that have failed in some test cases

1) Combine:-

It fails when the two ranges aren't null .
The output must be the minimum of the 2 lower ranges and the maximum of 2 upper ranges but it returns the be the minimum of the 2 lower ranges and also the minimum of 2 upper ranges.

Like this test case:

```
Range ObjIntPositive1 =new Range(4,30);
Range ObjIntPositive2=new Range(1,20);
```

```
Assert.assertEquals(new
Range(1,30),Range.combine(ObjIntPositive1,ObjInt
Positive2));
```

The expected output =[1,30] but this case failed as the actual output=[1,20]
It doesn't matter if the range is positive, negative, integer or double , it returns the be the minimum of the 2 lower ranges and also the minimum of 2 upper ranges.

2) Contains:-

It fails when the Value I passed to the function is the lower Bound of the Range.
That means during Implementation of this function ,the developer check if this (value > Lower Bound && value <= Upper Bound)
So it fails in lower bound as he considered the range is opened in the lower and closed on the upper like this]lower, upper]. But it' closed in the two Bounds like this [lower, upper] so he should check if (value >= Lower Bound && value <= Upper Bound).

Like this test case:

```
Assert.assertTrue(ObjIntPositive1.contains(4));
```

If must be true but it fails in this Test Case even if the lower Bound is positive, negative, integer or double.

3) Expand:-

It fails in the value of Lower Bound as expand mean to increase the length of the range so that the value of lower Bound must be decrease and the Upper Bound must be increase so the equation is:
New Lower Bound= old Lower Bound – Lower Margin * Length
New Upper Bound= old Upper Bound + Upper Margin * Length

But this implemented in the function for lower Bound like This:
New Lower Bound= old Lower Bound + Lower Margin * Length

Test Case:

```
Range obj =new Range(2,6);
Assert.assertEquals(new
Range(1,8),Range.expand(obj,0.25,0.5));
```

Expected =[1,8] but the actual =[3,8] so it fails even if the Range Bounds are positive, negative, integer or double

And it also fails when the Range is Null as it must throw exception with according to its documentation

```
InvalidParameterException
```

But the function throw other exception which is

```
 InvalidArgumentException
```

Test Case:

```
@Test(expected =
InvalidParameterException.class)
public void TestExandNull()
{
    Range.expand(null,0.25,0.5);
}
```

But when I searched for nullNotPermitted() ,I found that it throw the IllegalArgumentException not InvalidParameterException and in this function we used the nullNotPermitted() so it may be a Bug from Java itself not Test Case fail.

4) Intersect:-

It fails as it returns true in all cases even if the range passed dosen't intersect with the Range.

Test Case:-

```
Range ObjIntPositive1 =new Range(4,30);
Assert.assertFalse(ObjIntPositive1.intersects(0,
3));
```

The Test Case fails as the call of the function returns true not false.

Another Test Case:-

```
Range ObjDoubleNegative1=new Range(-12.5,-7.9);
Assert.assertFalse(ObjDoubleNegative1.intersects
(-3.5,-1.9));
```

The Test Case fails as the call of the function returns true not false.

So if fails if the Range Bounds are > the upper bound of the original range Or < the Lower Bound of the original bound.

➔ For null not permission in Two Shift Functions: -

1) Shift(Range , delta)

it fails when the Range is Null as it must throw exception with according to its documentation

```
InvalidParameterException
```

But the function throw other exception which is

```
 InvalidArgumentException
```

Test Case: -

```
@Test(expected =
InvalidParameterException.class)
public void TestShiftNull()
{
    Range.shift(null,3);
}
```

But when I searched for nullNotPermitted() ,I found that it throw the IllegalArgumentException not InvalidParameterException and in this function we used the nullNotPermitted() so it may be a Bug from Java itself not Test Case fail.

2) Shift(Range, Delta, allowZeroCrossing)
   it fails when the Range is Null as it must throw exception with according to its documentation

```
InvalidParameterException
```

   But the function throw other exception which is

```
InvalidArgumentException
```

   Test Case: -

```
@Test(expected =
InvalidParameterException.class)
public void TestShiftNull()
{
    Range.shift(null,3);
}
```

   But when I searched for nullNotPermitted() ,I found that it throw the IllegalArgumentException not InvalidParameterException and in this function we used the nullNotPermitted() so it may be a Bug from Java itself not Test Case fail.