Lab 6 Grading

•3 points are received maximum for all the TA's unit tests passing.

•30 TA tests total. Points = 3 * (*# tests passes)/* 30 rounded to the nearest half point.

•The 4th point is received for having 20+ meaningful tests with comments of your own.

TA Unit tests:

```
@Test
public void testConstructor1(){
     boolean error=false;
     try{
        Fraction f = new Fraction(9,0);//should give error
     }catch (IllegalArgumentException e){
          error = true;
     }
     assertTrue(error);
}
@Test
public void testContructor2(){
     boolean error=false;
     try{
          Fraction f = new Fraction(9,1);//should not give error
     }catch (IllegalArgumentException e){
          error = true;
     }
     assertFalse(error);
}
@Test
public void testContructor3(){
     boolean error=false;
     try{
          Fraction f = new Fraction(0);//should not give error
     }catch (Exception e){
          error = true;
     }
     assertFalse(error);
}
@Test
public void testCompare1(){
     Fraction f1 = new Fraction(1,2);
     Fraction f2 = new Fraction(1,2);
     assertEquals(0,f1.compareTo(f2));//1/2 should equal 1/2
}
@Test
public void testCompare2(){
     Fraction f1 = new Fraction(1,2);
     Fraction f2 = new Fraction(1,4);
     assertTrue(f1.compareTo(f2)>0);// 1/2 > 1/4
}
@Test
public void testCompare3(){
```

```
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,4);
    assertTrue(f2.compareTo(f1)<0);// 1/2 < 1/4
}
@Test
public void testadd1(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,2);
    Fraction f3 = new Fraction(1);
    f1.add(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2+1/2 = 1
}
@Test
public void testadd2(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,3);
    Fraction f3 = new Fraction(5,6);
    f1.add(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2+1/3 = 5/6
}
@Test
public void testsub1(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,2);
    Fraction f3 = new Fraction(0);
    f1.subtract(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2-1/2 = 0
}
@Test
public void testsub2(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,3);
    Fraction f3 = new Fraction(1,6);
    f1.subtract(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2-1/3 = 1/6
}
@Test
public void testmul1(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,2);
    Fraction f3 = new Fraction(1,4);
    f1.multiply(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2 * 1/2 = 1/4
}
@Test
public void testmul2(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(0);
    Fraction f3 = new Fraction(0);
    f1.multiply(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2 * 0 = 0
}
@Test
public void testdiv1(){
    Fraction f1 = new Fraction(1,2);
    Fraction f2 = new Fraction(1,2);
    Fraction f3 = new Fraction(1);
    f1.divide(f2);
    assertEquals(0,f1.compareTo(f3));// 1/2 / 1/2 = 1
}
```

```java
@Test
public void testdiv2(){
     Fraction f1 = new Fraction(1,2);
     Fraction f2 = new Fraction(1,3);
     Fraction f3 = new Fraction(3,2);
     f1.divide(f2);
     assertEquals(0,f1.compareTo(f3));// 1/2 / 1/3 = 3/2
}
@Test
public void testdiv3(){
     Fraction f1 = new Fraction(1,2);
     Fraction f2 = new Fraction(0);
     boolean error=false;
     try{
          f1.divide(f2);//should give error
     }catch ( Exception e){
          error = true;
     }
     assertTrue(error);
}
@Test
public void testpow1(){
     Fraction f1 = new Fraction(1,2);
     Fraction f3 = new Fraction(1,4);
     f1.pow(2);
     assertEquals(0,f1.compareTo(f3));// 1/2 ^ 2 == 1/4
}
@Test
public void testpow2(){
     Fraction f1 = new Fraction(1,2);
     Fraction f3 = new Fraction(1,8);
     f1.pow(3);
     assertEquals(0,f1.compareTo(f3));// 1/2 ^ 3 == 1/8
}
@Test
public void testpow3(){
     Fraction f1 = new Fraction(1,2);
     Fraction f3 = new Fraction(2);
     f1.pow(-1);
     assertEquals(0,f1.compareTo(f3));// 1/2 ^ -1 == 2
}
@Test
public void testpow4(){
     Fraction f1 = new Fraction(1,2);
     Fraction f3 = new Fraction(1);
     f1.pow(0);
     assertEquals(0,f1.compareTo(f3));// 1/2 ^ 0 == 1
}
@Test
public void testReduce1() {
     Fraction f1 = new Fraction(1, 2);
     f1.reduce();
     Fraction f3 = new Fraction(1, 2);
     assertEquals(0,f1.compareTo(f3));// 1/2 reduces should be itself
}
@Test
public void testReduce2(){
     Fraction f1 = new Fraction(2,4);
     f1.reduce();
     Fraction f3 = new Fraction(1,2);
     assertEquals(0,f1.compareTo(f3));// 2/4 => 1/2
```

```
}
@Test
public void testReduce3(){
     Fraction f1 = new Fraction(12,15);
     f1.reduce();
     Fraction f3 = new Fraction(4,5);
     assertEquals(0,f1.compareTo(f3));// 12/15 => 4/5
}
@Test
public void testtoString1(){
     Fraction f = new Fraction(1);
     assertEquals("1", f.toString());
}
@Test
public void testtoString2(){
     Fraction f = new Fraction(1,1);
     assertEquals("1",f.toString());
}
@Test
public void testtoString3(){
     Fraction f = new Fraction(1,2);
     assertEquals("1/2",f.toString());
}
@Test
public void testtoString4(){
     Fraction f = new Fraction(2,4);
     assertEquals("1/2",f.toString());
}
@Test
public void testtoString5(){
     Fraction f = new Fraction(-2,-4);
     assertEquals("1/2",f.toString());
}
@Test
public void testtoString6(){
     Fraction f = new Fraction(-2,4);
     assertEquals("-1/2",f.toString());
}
@Test
public void testtoString7(){
     Fraction f = new Fraction(2,-4);
     assertEquals("-1/2",f.toString());
}
@Test
public void testtoString8(){
     Fraction f = new Fraction(0,2);
     assertEquals("0",f.toString());
}
```