```
/**
 * This is my code! It's goal is to form polynomials
 * CS 312 - Assignment 5
 * @author Gil Carlson
 */

import java.util.List;
import java.util.LinkedList;
import java.util.ArrayList;
import java.util.Collections;

class Polynomial
{
    public List<?> createPolynomial( )
    {
                LinkedList<Terms> a = new LinkedList<Terms>();

        a.add( new Terms( 5, 2 ));
                a.add( new Terms( 2, 5 ));
                a.add( new Terms( 3, 4 ));

                Collections.sort(a);


        return a;
    }

    public int degree( List<Terms> b )
    {
        int degree = 0;

        for( Terms y: b )
                {
                     if (y.termExponent > degree)
             {
                 degree = y.termExponent;
             }
                }

        return degree;

    }

    public LinkedList<Terms> add( LinkedList<Terms> poly1, LinkedList< Terms> po
ly2 )
    {
        LinkedList<Terms> addedPolynomial = new LinkedList<Terms>();
        List<Terms> tempPoly2 = poly2;

        for( Terms y: poly1 )
                {
             addedPolynomial.add( y );

             for( Terms z: tempPoly2 )
             {
                        if ( y.termExponent == z.termExponent)
                        {
                            addedPolynomial.add( new Terms( y.termBase + z.termB
ase, y.termExponent ));
                       addedPolynomial.remove( y );
                   tempPoly2.remove( z );
                        }

             }
                }

        for( Terms d : tempPoly2 )
        {
            addedPolynomial.add( d );
```



```
        }

        Collections.sort( addedPolynomial );

        System.out.println( );

        for( Terms c: addedPolynomial )
                {
                        System.out.print( c.termBase + "x" +  c.termExponent + "
+" );
                }

        return addedPolynomial;

    }

    public LinkedList<Terms> multiply( LinkedList<Terms> poly1, LinkedList<Terms
> poly2 )
    {
         LinkedList<Terms> multiplyPolynomial = new LinkedList<Terms>();

                for( Terms y: poly1 )
                {

                     for( Terms z: poly2 )
                     {
                                multiplyPolynomial.add( new Terms( y.termBase *
z.termBase, y.termExponent + z.termExponent ));

                     }
                }

                Collections.sort( multiplyPolynomial );

        System.out.println( );

                for( Terms c: multiplyPolynomial )
                {
                        System.out.print( c.termBase + "x" +  c.termExponent + "
+" );
                }

        return multiplyPolynomial;

    }

}
```

Oct 29, 20 14:10 **Tester.java** Page 1/1

```
/**
 * This is my code! It's goal is to test the program
 * CS 312 - Assignment 5
 * @author Gil Carlson
 */

import java.util.*;

public class Tester
{
    public static void main( String [] args )
    {

                LinkedList<Terms> a = new LinkedList<Terms>();
        LinkedList<Terms> b = new LinkedList<Terms>();


                a.add( new Terms( 5, 2 ));
        b.add( new Terms( 7,2 ));
        b.add( new Terms( 7,4 ));
        a.add( new Terms( 5,6 ));
                a.add( new Terms( 3, 4 ));
        a.add( new Terms( 4,5 ));

                Collections.sort(a);

        for( Terms y: a )
        {
            System.out.print( y.termBase + "x" + y.termExponent + " ");
        }

        for( Terms z: b )
                {
                        System.out.println("\n" + z.termBase + "x" + z.termExpon
ent);
                }


        Polynomial poly = new Polynomial();
        System.out.println( "Degree is " + poly.degree(a ));
        System.out.println( "Degree is " + poly.degree(b ));

        for(int i=0; i<K1; i++)
            poly = poly + poly;
        for(int i=0; i<K2; i++)
            poly = poly * term;
        for(int i=0; i<K3; i++)
            poly = poly * poly;

        poly.add( a, b );
        poly.multiply( a, b );
    }
}
```

Oct 28, 20 12:43 **Terms.java** Page 1/1

```
/**
 * This is my code! It's goal is to create and store terms
 * CS 312- Assignment 5
 * @author Gil Carlson
 */

class Terms implements Comparable<Terms>
{
    int termExponent;
    int termBase;


    Terms( int termBase, int termExponent )
    {
        this.termBase = termBase;
        this.termExponent = termExponent;
    }

public int compareTo( Terms exp )
{
    if( termExponent == exp.termExponent)
        return 0;
    else if( termExponent > exp.termExponent)
        return -1;
    else
        return 1;
}
}
```