Module 9

Aakash Shah
Team 4: Pause&Ponder*

April 1, 2018

1 Linear combinations of functions: examples

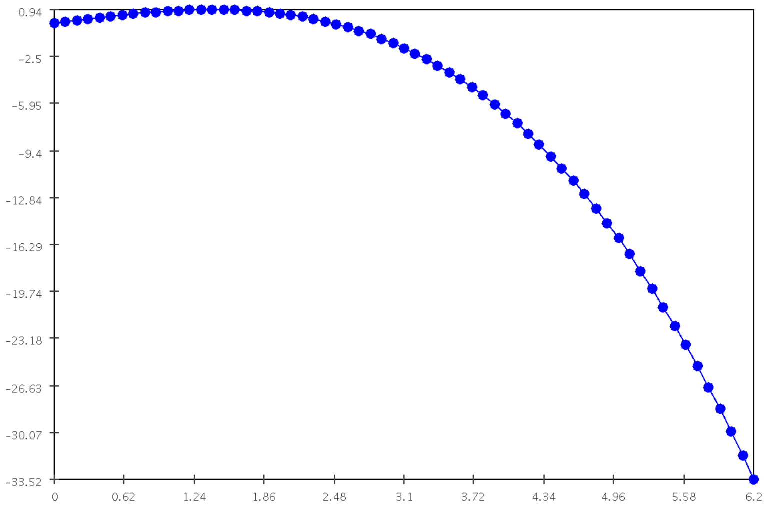
1.1 In-Class Exercise 1

Download LinCombExample.java and verify that g(x) is being computed as above. Compile and execute to draw the function g. You will also need Function.java and SimplePlotPanel.java

Yes it is correct.

```
public static void main (String[] argv)
{
    Function G = new Function ("g(x) vs x");
    for (double x=0; x<=6.283; x+=0.1) {
        G.add (x, g(x));
    }

    // Draw the curve.
    G.show ();
}

static double g (double x)
{
    // Array of coefficients (scalars in the linear combination).
    double[] alpha = {0, 1.0, 0, -0.16666666666666666};

    // Here, f_1(x) = x,  f_2(x) = 0,  f_3(x) = x^3.
    double g = alpha[1] * x  +  alpha[3] * Math.pow(x,3) ;

    return g;
}
```

*Team Member: Rohan Shetty

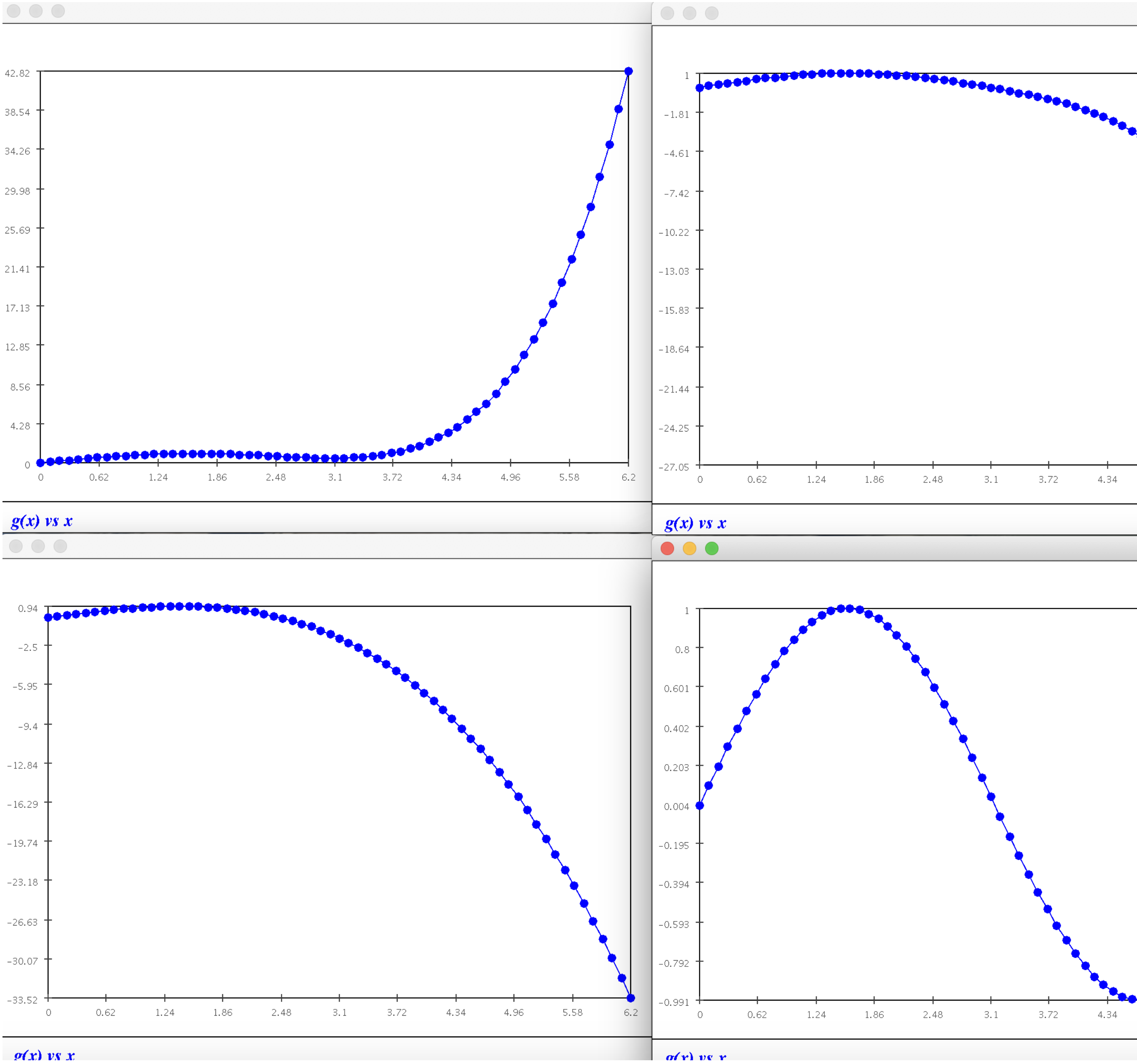


g(x) vs x

1.2 In-Class Exercise 2

Download LinCombExample2.java and verify that g(x) is being computed as above. Use different values of n in the program to include additional terms step by step. What is the function g(x) remind you of?

```
static double g (double x)
{
// Array of coefficients (scalars in the linear combination).
double[] alpha = {0, 1.0, 0, -0.16666666666666666, 0, 0.008333333333333333, 0, -1.9841269841269839E-4,

// How many terms to include: Start with n=3, then change to n=5,
// n=7, then n=13.
int n = 3;

double g = 0;
for (int k=0; k<=n; k++) {
    g = g + alpha[k] * Math.pow (x, k);
}

return g;
}
```



It is a sinusoidal (sin) function.

1.3 In-Class Exercise 3

Download LinCombExample3.java use different values of n to print the error between the approximation function g(x) and the function that g(x) seeks to approximate.

1) What is the difference in error between n=3 and n=13? What is the ratio?

2) How does the ratio differ when the interval is $[0, \pi]$ versus $[0, 2\pi]$? $[0, \pi 2]$?

3) Why is the error at each point x multiplied by deltaX, and what does this say about the totalError?

1) For n = 3, Total error: 44.039450550769644
and n = 13, Total error: 0.22858564290037903

Ratio : approx(192:1)

2) For $[0, \pi]$, Total error: 19.530699449230283
For $[0, 2\pi]$, Total error: 44.039450550769644
Ratio :

and

For $[0, \pi]$, Total error: 19.530699449230283
For $[0, \frac{\pi}{2}]$, Total error: 19.530699449230283
Ratio : 1

3) Why is the error at each point x multiplied by deltaX, and what does this say about the totalError?

1.4 In-Class Exercise 4

Look up the Taylor series for sin(x). Then, download SinTaylor.java and implement the factorial function. Compare the coefficients printed to those in the alpha[] array used in prior exercises.

```
public class SinTaylor {
        public static double lookup[];
    public static void main (String[] argv)
    {
                int range = 13;
                lookup = new double[range+1];                    //Initialized with 0;

                for (int k=1; k<=range; k+=2) {
                    double alpha = 1.0 / factorial (k);
                    System.out.println ("alpha (without the sign): " + alpha);
                }
    }
    static double factorial (int k)
    {
                if(k <= 1) {
                        return 1;
                }
                if(lookup[k] != 0.0) {
                        return lookup[k];
                }
                lookup[k] = k * factorial(k-1);
                return lookup[k];
    }
}
```

Values of alpha for the odd powers of sin(x) is :

alpha (without the sign): 1.0
alpha (without the sign): 0.16666666666666666
alpha (without the sign): 0.008333333333333333
alpha (without the sign): 1.984126984126984E-4
alpha (without the sign): 2.7557319223985893E-6
alpha (without the sign): 2.505210838544172E-8
alpha (without the sign): 1.6059043836821613E-10

Compared to the previous alphas :

alpha: 0.0
alpha: 1.0
alpha: 0.0
alpha: -0.16666666666666666
alpha: 0.0
alpha: 0.008333333333333333
alpha: 0.0
alpha: -1.9841269841269839E-4
alpha: 0.0
alpha: 2.7557319223985884E-6
alpha: 0.0
alpha: -2.5052108385441714E-8
alpha: 0.0
alpha: 1.605904383682161E-10

Since sin(x) is a odd function the taylor series just takes the odd powers.

The taylor series for sin(x) looks like :

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

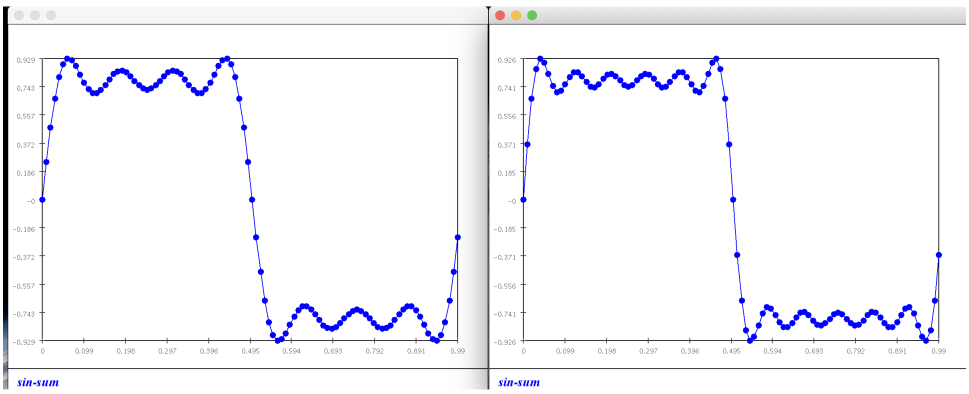
$$= \sum_{n=1}^{\infty}(-1)^{(n-1)} \frac{x^{2n-1}}{(2n-1)!} \overset{\text{or}}{=} \sum_{n=0}^{\infty}(-1)^{n} \frac{x^{2n+1}}{(2n+1)!}$$

1.5 In-Class Exercise 5

Download TrigPolyLinComb.java, which ends with k=3. Observe the result. Try adding additional terms until k=11.

```
public class TrigPolyLinComb {

    public static void main (String[] argv)
    {
                int k = 1;
                int range = 3;
                double y = 0.0;
                Function Fsum = new Function ("sin-sum");
                for (double x=0; x<=1; x+=0.01) {
                        y = 0.0;
                        for(k = 1;k <= range; k += 2) {
                            y += 1.0/k * Math.sin (2*Math.PI*k*x);
                        }
//                    double y = Math.sin (2*Math.PI*x)
//                               + 1.0/3.0 * Math.sin (2*Math.PI*3*x);
                    Fsum.add (x, y);
                }
                Fsum.show ();
    }

}
```

The graph for k = 7 and k = 11 looks like :



1.6 In-Class Exercise 6

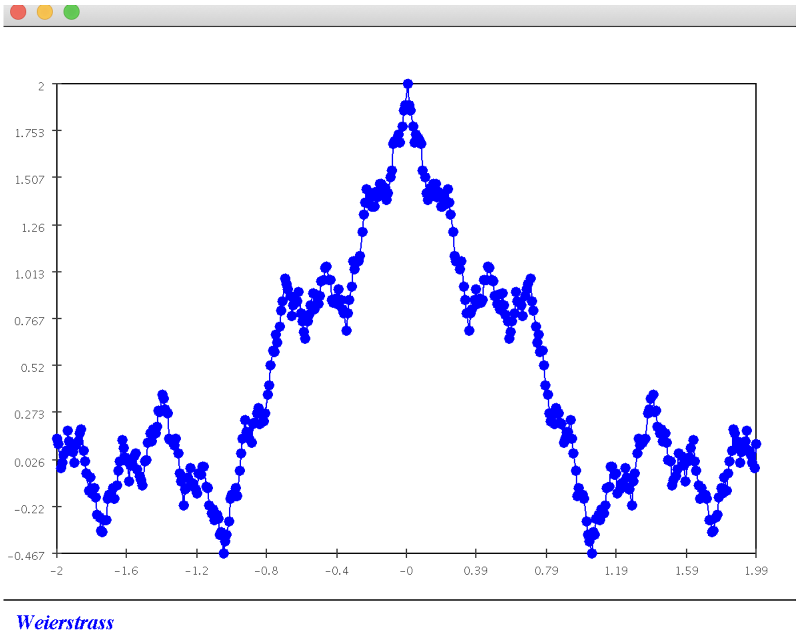
What is an example of a function h(x) that no linear combination of the different $\sin(2\pi kx)$ **functions could possibly approximate?**

h(x) $= \sin^2(2\pi kx) + \cos^2(2\pi kx)$

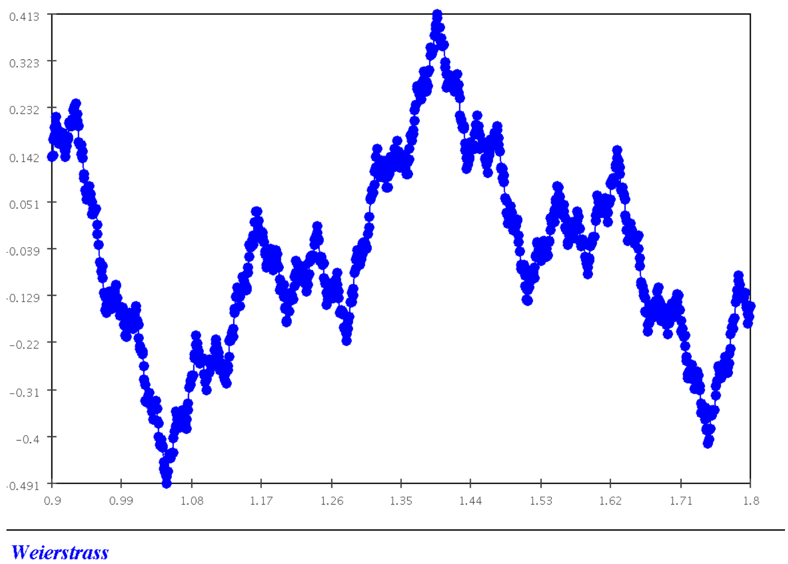
1.7 In-Class Exercise 7

Download Weierstrass.java and execute. Then change the interval as directed.

xLeft=-2, xRight=2, deltaX=0.01



xLeft=0.9, xRight=1.8, deltaX=0.001



2 Bernstein polynomials

2.1 In-Class Exercise 8

What does $\binom{n}{k}$ evaluate to in these four cases: k=0,k=1,k=n−1,k=n?

$\binom{n}{k}$, when k = 0 ; $\binom{n}{0} = 1$
$\binom{n}{k}$, when k = 1 ; $\binom{n}{1} = $ n
$\binom{n}{k}$, when k = n-1 ; $\binom{n}{n-1} = \binom{n}{1} = $ n
$\binom{n}{k}$, when k = n ; $\binom{n}{n} = \binom{n}{0} = 1$

2.2 In-Class Exercise 9

Look up the formula for $\binom{n}{k}$ and explain how it's derived. Write code in Combinations.java to compute it.

A k-combination of a set S is a subset of k distinct elements of S. If the set has n elements, the number of k-combinations is equal to the binomial coefficient

$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1}$

```
public class Combinations {

        static int lookup[];
    public static void main (String[] argv)
    {
        // Try k=0,1,2,3,4,5.
        int k = 1;

        int r = numCombinations (5,k);
        System.out.println (r);
    }

    static int numCombinations (int n, int k)
    {

                lookup = new int[n+1];
                int n1 = factorial(n);
                lookup = new int[k+1];
                int k1 = factorial(k);
                lookup = new int[(n-k)+1];
                int nk = factorial(n-k);
                return (n1/(k1 * nk));
    }

    static int factorial (int k)
    {
                if(k <= 1) {
                        return 1;
                }
                if(lookup[k] != 0) {
                        return lookup[k];
                }
```

```java
                lookup[k] = k * factorial(k-1);
                return lookup[k];

        }

}
```

2.3 In-Class Exercise 10

Prove that $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Write recursive code in CombinationsRecursive.java to compute it and plot the shape vs. k. Why is it symmetric? Look up Pascal's triangle and draw a few rows. Explain what the above result has to do with Pascal's triangle.

The reason $\binom{n}{k}$ is embedded in Pascal's triangle is that

$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$

and $\binom{n}{0} - \binom{n}{n} = 1$ for all n. Obviously because there is only one possible combination to choose nothing from something and everything from everything. So the numbers $\binom{n}{k}$ satisfy the defining rule of Pascal's triangle. The way to think about it : How many ways can you choose k items from a set of n, where one of the items is marked? Who cares if one is marked, the answer is $\binom{n}{k}$, but we can also say that $\binom{n-1}{k-1}$ choices which include the marked item and $\binom{n-1}{k}$ which exclude the marked item so the answer will be to add both subsets to finish the set.

A more refined way to think of it is that the values in Pascal triangle can be expressed as :

$n = 0$							1						
$n = 1$						1		1					
$n = 2$					1		2		1				
$n = 3$				1		3		3		1			
$n = 4$			1		4		6		4		1		
$n = 5$		1		5		10		10		5		1	
$n = 6$	1		6		15		20		15		6		1

$n = 0$							$\binom{0}{0}$						
$n = 1$						$\binom{1}{0}$		$\binom{1}{1}$					
$n = 2$					$\binom{2}{0}$		$\binom{2}{1}$		$\binom{2}{2}$				
$n = 3$				$\binom{3}{0}$		$\binom{3}{1}$		$\binom{3}{2}$		$\binom{3}{3}$			
$n = 4$			$\binom{4}{0}$		$\binom{4}{1}$		$\binom{4}{2}$		$\binom{4}{3}$		$\binom{4}{4}$		
$n = 5$		$\binom{5}{0}$		$\binom{5}{1}$		$\binom{5}{2}$		$\binom{5}{3}$		$\binom{5}{4}$		$\binom{5}{5}$	
$n = 6$	$\binom{6}{0}$		$\binom{6}{1}$		$\binom{6}{2}$		$\binom{6}{3}$		$\binom{6}{4}$		$\binom{6}{0}$		$\binom{6}{6}$

Since we know that a number is the sum of two numbers on the top, for example searching for $\binom{5}{2}$ we have to find the two numbers above it that is $\binom{4}{1}$ and $\binom{4}{2}$.

This is musical!

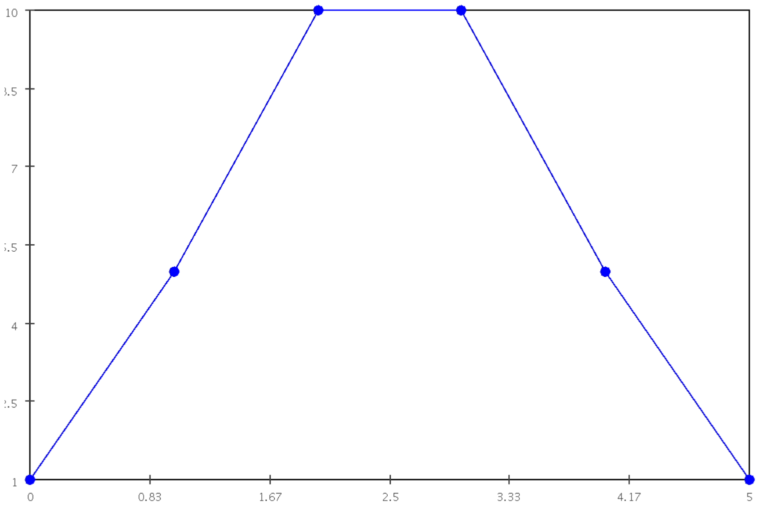
The following code for CombinationsRecursive using Pascal's triangle. There are 3 versions to decrease comparisons.

```java
import java.util.ArrayList;
```

```
import java.util.HashMap;

public class CombinationsRecursive {

static int lookup[];
static HashMap<ArrayList<Integer>,Integer> lookupTable;
public static void main (String[] argv){
                int n = 5;
                Function C = new Function ("n-choose-k vs k");
                for (int k=0; k<=n; k++) {
                        lookupTable = new HashMap<>();
                    int r = numCombinationsRecursive (n,k);
                    System.out.println("---------");
                    C.add (k, r);
                }
                C.show ();
}

static int numCombinationsRecursive (int n, int k){
                if ((n==k) || (n==1) || (k==0)) {
                        System.out.println("n: " + n + " k: " + k + " r: "+ 1);
                    return 1;
                }

                if(nkInMap(n,k)) {
                        return getValue(n,k);
                }

            ArrayList<Integer> list = new ArrayList<Integer>(2);
                list.add(n);
                list.add(k);
                lookupTable.put(list, numCombinationsRecursive(n-1,k)+numCombinations
                System.out.println("n: " + n + " k: " + k + " r: "+ lookupTable.get(li
                return lookupTable.get(list);
}

private static int getValue(int n, int k) {
                ArrayList<Integer> list = new ArrayList<Integer>(2);
                list.add(n);
                list.add(k);
                return lookupTable.get(list);
}

private static boolean nkInMap(int n, int k) {
                ArrayList<Integer> list = new ArrayList<Integer>(2);
                list.add(n);
                list.add(k);
                if(lookupTable.containsKey(list)) {
                        return true;
                }
                return false;
}
}
```

Output for $\binom{5}{2}$
n: 2 k: 2 r: 1
n: 1 k: 1 r: 1
n: 1 k: 0 r: 1
n: 2 k: 1 r: 2

n: 3 k: 2 r: 3
n: 2 k: 0 r: 1
n: 3 k: 1 r: 3
n: 4 k: 2 r: 6
n: 3 k: 0 r: 1
n: 4 k: 1 r: 4
n: 5 k: 2 r: 10



n-choose-k vs k

3 Proofs

3.1 In-Class Exercise 12

Prove that $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ and implement this approach as a second recursive method in CombinationsComparison2.java. Try $n = 10$ and $n = 20$. Why is the return type double? Implement an iterative version of the above tail recursion.

```
n=5       k=0      p=1      q=1      r=1.0     s=1.0
numCalls=12        numCallsRecursive=1         numCallsRecursive2=1      numIterations=0

n=5       k=1      p=5      q=5      r=5.0     s=5.0
numCalls=23        numCallsRecursive=10        numCallsRecursive2=3      numIterations=1

n=5       k=2      p=10     q=10     r=10.0    s=10.0
numCalls=34        numCallsRecursive=23        numCallsRecursive2=6      numIterations=3

n=5       k=3      p=10     q=10     r=10.0    s=10.0
numCalls=45        numCallsRecursive=36        numCallsRecursive2=10     numIterations=6

n=5       k=4      p=5      q=5      r=5.0     s=5.0
numCalls=56        numCallsRecursive=45        numCallsRecursive2=15     numIterations=10

n=5       k=5      p=1      q=1      r=1.0     s=1.0
numCalls=68        numCallsRecursive=46        numCallsRecursive2=16     numIterations=15


TotalnumCalls=68
TotalnumCallsRecursive=46
TotalnumCallsRecursive2=16
TotalnumIterations=15
```

```java
static double numCombinationsIterative (int n, int k)
{
            double result = 1.0;
            for(int i= 0;i<=n;i++) {
                    if((k-i) > 0) {
                            result *= ((n-i)*(1.0))/((k-i)*(1.0));
                            numIterations++;
                    }
            }
            return result;
}
```

```java
static double numCombinationsRecursive2 (int n, int k)
{
            numCallsRecursive2 ++;
        if(n==k || n==1 || k==0) {
                    return 1.0;
        }

        if(nkInDoubleMap(n,k)) {
                    return getDoubleValue(n,k);
        }
```

```
            ArrayList<Integer> list = new ArrayList<Integer>(2);
            list.add(n);
            list.add(k);
            double ans = (n * 1.0)/(k * 1.0);
            lookupNewTable.put(list, ans * numCombinationsRecursive2(n-1,k-1));
            return lookupNewTable.get(list);
    }
```

The return type is double so that the division can give the best approximate value of the division involved.

3.2 In-Class Exercise 16

For any two numbers a,b such that $a < b$ and $t \in [0,1]$ prove that $(1-t)a + tb \in [a,b]$. Then, use this fact and Proposition 9.1 to prove 9.2.

To prove : $b_{n,k}(t) \geq 0$ for all n,k and $t \in [0,1]$.

$b_{n,k}(t) = (1-t)b_{n-1,k}(t) + tb_{n-1,k-1}(t)$ we can use this recursive proposition to prove the above property.

Base case : It is easily seen that the functions $B_{0,1}(t) = 1-t$ and $B_{1,1}(t) = t$ are both non-negative for $0 \leq t \leq 1$.

Initial Hypothesis(IH) : If we assume that all Bernstein polynomials of degree less than k are non-negative, then by using the recursive definition of the Bernstein polynomial,

we can write $B_{i,k}(t) = (1-t)B_{i,k-1}(t) + tB_{i-1,k-1}(t)$ and argue that

$B_{i,k}(t)$ is also non-negative for $0 \leq t \leq 1$, since **all components on the right-hand side of the equation are non-negative** components for $0 \leq t \leq 1$.

By induction, all Bernstein polynomials are non-negative for $0 \leq t \leq 1$.

In this process, we have also shown that each of the Bernstein polynomials is positive when $0 < t < 1$.

3.3 In-Class Exercise 19

Prove the below assertion.

$t^k.(1-t)^{n-k} = t^k.(1-t)^{n+1-k} + t^{k+1}.(1-t)^{n+1-(k+1)}$

In terms of Pascals triangle all we need to prove is :

k-th term at level n = k-th term at level n+1 + (k+1)-st term at level n+1

This can be done through an induction proof.

Base case : It is easily seen that the equation is true

L.H.S. => $t^0.(1-t)^0 = 1$
R.H.S. => $t^0.(1-t)^{0+1-0} + t^{0+1}.(1-t)^{0+1-(0+1)}$

$= 1 - t + t = 1$

for $n = 0, k = 0$ and similarly for $n = 1, k = 0, k = 1$.

Initial Hypothesis(IH) : If we assume that all further equations are similarly equal for n=n+1 then by using the recursive definition of the Bernstein polynomial,

we can write $t^k.(1-t)^{i-k} = t^k.(1-t)^{i+1-k} + t^{k+1}.(1-t)^{i+1-(k+1)}$ and argue that

...
By induction, all

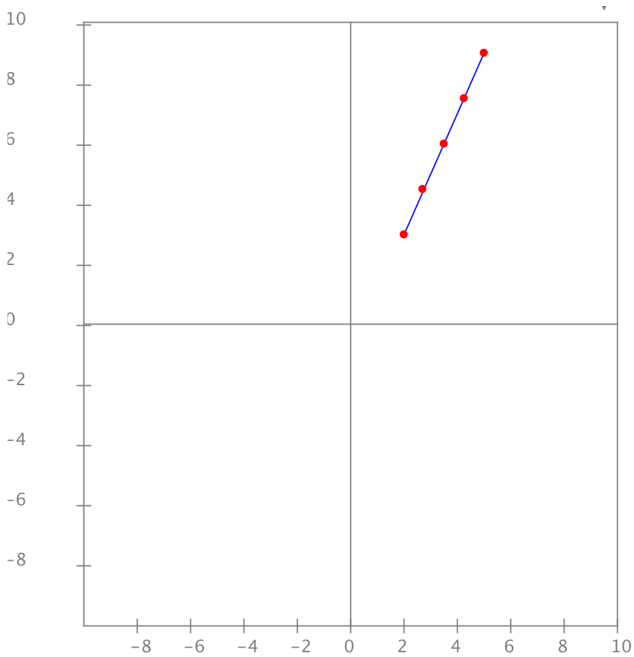
3.4 In-Class Exercise 25

Compile and execute DrawLineParametric.java. Notice that the computed points are always between the end points.

*** Prove that if x0<x1 then $x0 \leq x(t) \leq x1$ and if $x1 < x0, then x1 \leq x(t) \leq x0$. Obviously, the same will be true for $y(t)$.**

*** Prove that each point $(x(t), y(t))$ is on the line segment between the end points.**

*** What happens when $t < 0$ or $t > 1$? Experiment with your program, and then explain.**



1)
The parametric equation says that if there are two points (x_0, y_0),(x_1, y_1):

$x = x_1 +$ (change in x) $*t$
$y = y_1 +$ (change in y) $*t$

where t ranges from [0 to 1] where change in y is $y_1 - y_0$ and change in x is $x_1 - x_0$

When the points are being dealt in R^3 i.e. 3 dimensions on the real vector space, the only way to get an equation of line is the parametric equation.

Proof :

$(y - y_1) = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$

can be written as :

$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$

Since they are equal

$x - x_1 = (x_2 - x_1)t$
$\Rightarrow x = x_1 + (x_2 - x_1)t$

Similarly for $y(t)$.

2) We can prove this by the following example :

Parametrize the line segment that connects the points (2, 3) and (7, 9).

It's asking for a line segment. Taking the parameterization:

$x(t) = 2 + 5t$
$y(t) = 3 + 6t$

We know that when $t = 0$ we're at the point (2, 3) and when $t = 1$ we're at the point (7, 9). If we restrict the parameter so that $0 \leq t \leq 1$ then we find only the line segment that lies between those two points.

3) When the value of t goes less than 0 or more than 1 it extends the point in the direction of the line. Shown below.