Recursion Problems

- Factorial function :
  - recursive

```
public class Main {
    public static int fac(int x) {
        if (x == 0)
            return 1;
        return x*fac(x-1);
    }
    public static void main(String[] args) {
        int x = StdIn.readInt();
        System.out.println(fac(x));
    }
}
```

  - iterative

```
public class Main {
    public static int fac(int x) {
        int f = 1;
        for (int i = 2; i <= x; i++) {
            f *= i;
        }
        return f;
    }
    public static void main(String[] args) {
        int x = StdIn.readInt();
        System.out.println(fac(x));
    }
}
```

- Fibonacci

```
public class Main {
    public static int fib(int x) {
        if (x < 2)
            return 1;
        return fib(x-1)+fib(x-2);
    }
    public static void main(String[] args) {
        for (int i = 0; i < 16; i++) {
            System.out.print(fib(i) + " ");
        }
    }
}
```

- Euclidean GCD :
  - Repeated subtraction method

```
public class Main {
    public static int gcd(int x, int y) {
        if (x == y)
            return x;
        else if (x < y)
            return gcd(x, y-x);
        else
            return gcd(x-y, y);
    }
    public static void main(String[] args) {
        int x = StdIn.readInt(), y = StdIn.readInt();
        System.out.println(gcd(x, y));
    }
}
```

  - Remainder method

```
public class Main {
    public static int gcd(int x, int y) {
        if (y == 0)
            return x;
        return gcd(y, x%y);
    }
    public static void main(String[] args) {
        int x = StdIn.readInt(), y = StdIn.readInt();
        System.out.println(gcd(x, y));
    }
}
```

  - Iterative method

```
public class Main {
    public static int gcd(int x, int y) {
        int result = 1;
        for (int i = 2; i < Math.min(x, y); i++) {
            if (x % i == 0)
                result = i;
        }
        return result;
    }
    public static void main(String[] args) {
        int x = StdIn.readInt(), y = StdIn.readInt();
        System.out.println(gcd(x, y));
    }
}
```

- Tower of Hanoi

```java
public class Main {
    public static void hanoiTower(int n, char x, char y, char z) {
        if (n == 1)
            System.out.println("Move top disk from " + x + " to " + z);
        else {
            hanoiTower(n-1, x, z, y);
            hanoiTower(1, x, y, z);
            hanoiTower(n-1, y, x, z);
        }
    }
    public static void main(String[] args) {
        hanoiTower(3, 'A', 'B', 'C');
    }
}
```

```java
public static boolean isPrime(int n)
{
    if (n < 2) return false;
    for (int i = 2; i <= n/i; i++)
        if (n % i == 0) return false;
    return true;
}
```

absolute value of an int value

```
public static int abs(int x)
{
   if (x < 0) return -x;
   else       return  x;
}
```

absolute value of a double value

```
public static double abs(double x)
{
   if (x < 0.0) return -x;
   else         return  x;
}
```

primality test

```
public static boolean isPrime(int n)
{
   if (n < 2) return false;
   for (int i = 2; i <= n/i; i++)
      if (n % i == 0) return false;
   return true;
}
```

hypotenuse of a right triangle

```
public static double hypotenuse(double a, double b)
{  return Math.sqrt(a*a + b*b);  }
```

harmonic number

```
public static double harmonic(int n)
{
   double sum = 0.0;
   for (int i = 1; i <= n; i++)
      sum += 1.0 / i;
   return sum;
}
```

uniform random integer in [0, n)

```
public static int uniform(int n)
{  return (int) (Math.random() * n);  }
```

draw a triangle

```
public static void drawTriangle(double x0, double y0,
                                double x1, double y1,
                                double x2, double y2 )
{
   StdDraw.line(x0, y0, x1, y1);
   StdDraw.line(x1, y1, x2, y2);
   StdDraw.line(x2, y2, x0, y0);
}
```

Program 2.1.3 Coupon collector (revisited)

```
public class Coupon
{
   public static int getCoupon(int n)
   {  // Return a random integer between 0 and n-1.
      return (int) (Math.random() * n);
   }

   public static int collectCoupons(int n)
   {  // Collect coupons until getting one of each value
      // and return the number of coupons collected.
      boolean[] isCollected = new boolean[n];
      int count = 0, distinct = 0;
      while (distinct < n)
      {
         int r = getCoupon(n);
         count++;
         if (!isCollected[r])
            distinct++;
         isCollected[r] = true;
      }
      return count;
   }

   public static void main(String[] args)
   {  // Collect n different coupons.
      int n = Integer.parseInt(args[0]);
      int count = collectCoupons(n);
      StdOut.println(count);
   }
}
```

n	*# coupon values (0 to n-1)*
isCollected[i]	*has coupon i been collected?*
count	*# coupons collected*
distinct	*# distinct coupons collected*
r	*random coupon*

```
% java Coupon 1000
6522
% java Coupon 1000
6481
```

```
% java Coupon 10000
105798
% java Coupon 1000000
12783771
```

```
public static void exchange(String[] a, int i, int j)
{
   String temp = a[i];
   a[i] = a[j];
   a[j] = temp;
}
```

```
public static void shuffle(String[] a)
{
   int n = a.length;
   for (int i = 0; i < n; i++)
      exchange(a, i, i + uniform(n-i));
}
```

find the maximum of the array values

```
public static double max(double[] a)
{
   double max = Double.NEGATIVE_INFINITY;
   for (int i = 0; i < a.length; i++)
      if (a[i] > max) max = a[i];
   return max;
}
```

dot product

```
public static double dot(double[] a, double[] b)
{
   double sum = 0.0;
   for (int i = 0; i < a.length; i++)
      sum += a[i] * b[i];
   return sum;
}
```

exchange the values of two elements in an array

```
public static void exchange(String[] a, int i, int j)
{
   String temp = a[i];
   a[i] = a[j];
   a[j] = temp;
}
```

print a one-dimensional array (and its length)

```
public static void print(double[] a)
{
   StdOut.println(a.length);
   for (int i = 0; i < a.length; i++)
      StdOut.println(a[i]);
}
```

read a 2D array of `double` values (with dimensions) in row-major order

```
public static double[][] readDouble2D()
{
   int m = StdIn.readInt();
   int n = StdIn.readInt();
   double[][] a = new double[m][n];
   for (int i = 0; i < m; i++)
      for (int j = 0; j < n; j++)
         a[i][j] = StdIn.readDouble();
   return a;
}
```

[Java exams solutions.pdf - Google Drive]

```
   public static double harmonic(int n)
   {
      if (n == 1) return 1.0;
      return harmonic(n-1) + 1.0/n;
   }

public class Euclid
{
   public static int gcd(int p, int q)
   {
      if (q == 0) return p;
      return gcd(q, p % q);
   }
   public static void main(String[] args)
   {
      int p = Integer.parseInt(args[0]);
      int q = Integer.parseInt(args[1]);
      int divisor = gcd(p, q);
      StdOut.println(divisor);
   }
}

   public class TowersOfHanoi
   {
      public static void moves(int n, boolean left)
      {
         if (n == 0) return;
         moves(n-1, !left);
         if (left) StdOut.println(n + " left");
         else      StdOut.println(n + " right");
         moves(n-1, !left);
      }
      public static void main(String[] args)
      {  // Read n, print moves to move n discs left.
         int n = Integer.parseInt(args[0]);
         moves(n, true);
      }
   }
```

```
   public static double harmonic(int n)
   {
      return harmonic(n-1) + 1.0/n;
   }
public static long fibonacci(int n)
{
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fibonacci(n-1) + fibonacci(n-2);
}
```

1- Write a program that reads in integers (as many as the user enters) from standard input and prints out the maximum and minimum values.

2- Write a program that takes an integer N from the command line, reads N double values from standard input, and prints their mean (average value) and standard deviation (square root of the sum of the squares of their differences from the average, divided by N−1).

3- Write a program that reads in text from standard input and prints out the number of words in the text. For the purpose of this exercise, a word is a sequence of non-whitespace characters that is surrounded by whitespace.

4- Write a program that takes a command-line argument N and plots an N-by-N checkerboard with red and black squares. Color the lower left square red.

Q.1 Write a program that takes an integer N from the command line, reads
N double values from standard input, and prints their mean (average value) and standard deviation (square root of the sum of the squares of their differences from the average, divided by N-1).

```
public class Main {
    public static void main(String[] args) {
        int N = StdIn.readInt();
        double arr[] = new double[N];
        double avrg = 0;
        for (int i = 0;i<N;i++)
        {
            arr[i] = StdIn.readDouble();
            avrg +=arr[i];
        }
        double sum2 = 0;
        for (int i = 0;i<N;i++)
        {
            sum2 +=Math.pow((arr[i]-(avrg/N)),2);
        }
        double ans = Math.sqrt(sum2/(N-1));
        StdOut.println(ans);
    }
}
```

Q.2 Write a program that reads in text from standard input and prints out the number of words in the text. For the purpose of this exercise, a word is a sequence of non-whitespace characters that is surrounded by whitespace.

```
public class Main {

    public static void main(String[] args) {
        String s = StdIn.readLine();
        int cntr = 0;
        for (int i = 0;i<s.length();i++)
        {
            if( s.charAt(i) == ' ')
                cntr++;
        }
        StdOut.println(s.length()-cntr);
    }}
```

Q.3 Write a program that takes a command-line argument N and plots an N-by-N checkerboard with red and black squares. Color the lower left square red.

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

        int n = Integer.parseInt(args[0]);
        StdDraw.setXscale(0, n);
        StdDraw.setYscale(0, n);

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (((i + j) % 2) != 0)
                    StdDraw.setPenColor(StdDraw.BLACK);
                else
                    StdDraw.setPenColor(StdDraw.RED);

                StdDraw.filledSquare(i + 0.5, j + 0.5, 0.5);
            }
        }
    }
}
```

Sheet (6):

1- Write a static method odd() that takes three boolean inputs and returns true if an odd number of inputs are true, and false otherwise.

2- Write a static method majority() that takes three boolean arguments and returns true if at least two of the arguments have the value true, and false otherwise. Do not use an if statement. Solution: here are two solutions: the first is concise; the second strictly adheres to the rules.

3- Write a static method eq() that takes two arrays of integers as arguments and returns true if they contain the same number of elements and all corresponding pairs of elements are equal, and false otherwise.

4- Write a static method lg() that takes a double value N as argument and returns the base 2 logarithm of N. You may use Java's Math library.

5- Consider the static method cube() below.

```
public static void cube(int i) {
  i = i * i * i;
}
```

How many times is the following for loop iterated?

```
for (int i = 0; i < 1000; i++)
   cube(i);
```

6- Write a method that takes an array of double values as argument and rescales the array so that each element is between 0 and 1 (by subtracting the minimum value from each element and then dividing each element by the difference between the minimum and maximum values). Use the max() method defined in the table in the text, and write and use a matching min() method.

1- Write a static method lg() that takes a double value N as argument and returns the base 2 logarithm of N. You may use Java's Math library.

```
public static double lg(double a)
{
    return Math.log(a)/Math.log(2);
}
```

2- Consider the static method cube() below.

```
public static void cube(int i) {
  i = i * i * i;
}
```

How many times is the following for loop iterated?

```
for (int i = 0; i < 1000; i++)
   cube(i);
```

- Ans : this function is void and parameters are passed by reference by default in Java, so the value of i doesn't change throughout the loop.

3- Write a method that takes an array of double values as argument and rescales the array so that each element is between 0 and 1 (by subtracting the minimum value from each element and then dividing each element by the difference between the minimum and maximum values). Use the max() method defined in the table in the text, and write and use a matching min() method.

```
Import jave.util.Arrays;
public static double[] scale(double a[],double min, double
max)
{
    for(int i=0;i<a.length;i++)
    {
        a[i]=(a[i]-min/(max-min));
    }
    return a;

}
```

1-Write and test a recursive function that returns the sum of the squares of the first n positive integers.

2- Write and test a recursive function that returns the sum of the first n elements of an array.

3- Write and test a recursive function that returns the power of n

4- Write and test a recursive function that returns the integer binary logarithm of an integer n (i.e., the number of times n can be divided by 2).

5- Trace the recursive implementation of the Euclidean Algorithm (Example 9.13 on page 171) on the call gcd(385, 231).