DNHI Homework 2
Recursion

Problem 1

Part A  Write an iterative method that computes a value of \( x^n \) for a positive integer \( n \) and a real number \( x \).

Part B  Write a recursive method that computes a value of \( x^n \) for a positive integer \( n \) and a real number \( x \).

Problem 2

Consider the following recursive method

```java
public int recMethod(int number) {
    if (number <= 0)
        return 0;
    if (number % 2 == 0)
        return recMethod(number - 1);
    else
        return number + recMethod(number - 1);
}
```

Part A
How many times is this method called (including the initial call) when we run `recMethod(10)`?

How many times is this method called (including the initial call) when we run `recMethod(-10)`?

Part B
What does `recMethod` do (i.e. what does it compute)?

Problem 3

Write a recursive method to compute the following series:

\[
\frac{1}{3} + \frac{2}{5} + \frac{3}{7} + \frac{4}{9} + \ldots + \frac{i}{2i+1}.
\]

Problem 4

Write a recursive method that computes the sum of the digits in an integer. Use the following method header:

```java
public static int sumOfDigits(long n)
```

For example, `sumOfDigits(234)` should return 9 (since \( 2 + 3 + 4 = 9 \)) and `sumOfDigits(390)` should return 12 (since \( 3 + 9 + 0 = 12 \)).

Problem 5

For each of the following recursive methods, rewrite it using iterations instead of recursion. HINT: in order to do so you should first figure out what these methods do.
Part A

```java
public int recur( int n ) {
    if (n < 0 ) throw new IllegalArgumentException ("negative argument detected");
    return recur_proper(n);
}

public int recur_proper( int n ) {
    if (n < 0 )
        return -1;
    else if ( n < 10 )
        return 1;
    else
        return ( 1 + recur_proper ( n / 10 ) );
}
```

Part B

```java
public int recur2 ( int n ){
    if (n < 0 )
        return -1;
    else if ( n < 10 )
        return n;
    else
        return ( n % 10 + recur2 ( n / 10 ) );
}
```

Problem 6

What would be printed by the following programs

Part A)

```java
public class CatsAndDogs {

    public static void main(String[] args) {
        foo("Cats and Dogs", 4);
    }

    public static void foo ( String s, int n ) {
        if (n <= 1)
            System.out.println("Cats");
        else {
            System.out.println( s ) ;
            foo ( s, n-1 ) ;
        }
    }
}
```

Part B)
```java
public class Numbers {

    public static void main(String[] args) {
        int[] list = {1, 2, 3, 4, 5};
        System.out.println( foo(list, 0, list.length-1) );
    }

    public static int foo ( int[] nums, int begin, int end ) {
        if ( begin == end )
            return nums[begin];
        else
            return nums[begin] + foo(nums, begin+1, end);
    }
}
```

### Problem 7

**Part A** Write a method that generates all sequences of a given length that contain digits 0 through 9 (all ten digits are allowed, repetitions are allowed)? Given length of the sequence equal to $n$, how many possible sequences are there?

**Part B** Modify the above method so that none of the generated sequences start with zero. How many of those sequences exist, given the length of $n$ digits?