DNHI Homework 2 Solutions
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 $x$.

Answer: The return value of -1 indicates an error condition.

```java
public static double powerIter (double x, int n) {
    double result = 1;
    if (n < 0 ) return -1;
    while (n > 0) {
        result = result * x;
        n--;
    }
    return result;
}
```

Part B  Write a recursive method that computes a value of $x^n$ for a positive integer $n$ and a real number $x$. Answer: The return value of -1 indicates an error condition.

```java
public static double powerRecursive (double x, int n) {
    if (n < 0 ) return -1;
    if (n == 0) return 1;
    return x * powerRecursive (x, n - 1);
}
```

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) ?

Answer: Called 11 times.

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

Answer: Called 1 time.
Part B
What does recMethod do (i.e. what does it compute)?
Answer: It computes the sum of odd numbers from zero to number.

Problem 3

Write a recursive method to compute the following series:
\[
\frac{1}{3} + \frac{2}{5} + \frac{3}{7} + \frac{4}{9} + \cdots + \frac{i}{2i+1}.
\]
Answer: The crucial part in this code is casting to double so that the fractions do not become all zero. Other than that it should be a straight forward implementation of the recursive method.

```java
public static double summation ( int num ) {
    // base case
    if ( num <= 0 ) return 0;
    // recursion
    return (double)num/(2*num+1) + summation(num-1);
}
```

Problem 4

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

```
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 ).

Answer: A possible solution could be:

```java
public static int sumOfDigits ( long n ) {
    // base case is when the number is zero
    if( n==0 ) return 0;
    //recursive case
    return ( (int)(n%10) + sumOfDigits( n/10 ) );
}
```

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 )
```
The code above computes the number of digits in the parameter $n$.

```java
public int recur (int n) {
  if (n < 0 ) throw new IllegalArgumentException("negative argument detected");
  if (n == 0 ) return 1;
  int solution = 0;
  while (n > 0 ){
    solution++;
    n = n/10;
  }
  return solution;
}
```

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 |
```
```java
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);
}
```

**Answer:**
The `foo` method computes the sum of the values in the list from between index `begin` and index `end`. So in this case it computes the sum of all elements in the list. It prints 15

### 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?

**Answer:**
With length of $n$ digits, the number of possible sequences is equal to $10^n$, for example, with length of $n = 4$, we have 10,000 different sequences.
```java
19 else { //add the next digit to the sequence (two possibilities)
20 for (int i = 0; i < 10; i++) {
21     //add digit i to the current sequence
22     getAllDecimalSequences( length, seq + Integer.toString(i));
23 }
24 }
25 }
26 }

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?

Answer: With this restriction, we only have 9 possibilities for the first digit and 10 for all the remaining digits. So there will be total of \( 9 \times 10^{n-1} \) sequences of length \( n \) that do not start with a zero.

```