

Introduction to Computer Programming, Spring Term 2017
Practice Assignment 4

Discussion: 11.3.2017 - 16.3.2017

Exercise 4-1 Power

Write a program that implements the definition of *power*, where *m* and *n* are integers entered by the user, such that *m* is the base and *n* is the exponent:

$$\text{power}(m,n) = m^n$$

Write a main to test your method.

Exercise 4-2 Euler

Write a method `Euler` to calculate the value of the mathematical constant *e* which is defined as:

$$e = \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

Implement first the method `factorial` ($\text{factorial}(n) = n!$) Write a main method to test your method.

Exercise 4-3 Fibonacci

To be discussed in the lab

The Fibonacci numbers are defined as follows. The zeroth Fibonacci number is 0. The first Fibonacci number is 1. The second Fibonacci number is $1 + 0 = 1$. The third Fibonacci number is $1 + 1 = 2$. In other words, except for the first two numbers each Fibonacci number is the sum of the two previous numbers.

Thus, Fibonacci Numbers, or “How many rabbits will you get after a year?” is given by

$$\begin{aligned} \text{Fib}(0) &= 0 \\ \text{Fib}(1) &= 1 \\ \text{Fib}(n) &= \text{Fib}(n-1) + \text{Fib}(n-2) \end{aligned}$$

Write a Java program `Fibonacci.java` that computes the *n*th Fibonacci number:

- Write a method `fib` to calculate the *n*th Fibonacci number.
- Add a `main` method that asks the user for *n* and displays the result, i.e. the *n*th Fibonacci number.

For more information on the use of Fibonacci Numbers please see Dr. Ron Knott’s excellent site at

<http://www.ee.surrey.ac.uk/Personal/R.Knott/Fibonacci/fib.html>

Exercise 4-4 Maximum

Given the following skeleton for your program, use the concept of overloading to define three versions of the method `max`, which should work as follows:

- The first method should find the maximum of two integers and return it.
- The second method should calculate the maximum of two floating-point numbers and return it.
- The third method should compare two strings and return the one that would appear later in the dictionary (lexicographical order). **Hint:** The method `x.compareTo(y)`, where `x` and `y` are strings, returns an integer greater than zero if the string `x` appears later than `y` in the dictionary.

```
public class Max
{
    public static void main(String args[])
    {
        System.out.println( max(1,5) );
        System.out.println( max(1.5,5.5) );
        System.out.println( max("Hello","World") );
    }
}
```

Exercise 4-5 Palindrome

To be discussed in the lab

Write a java program that determines whether the text the user inputs is a **palindrome** or not. A palindrome is a piece of text that can be read the same way in either direction (left to right and right to left). Examples of palindromes include words such as `racecar` and `noon`.

Exercise 4-6 Prime

Write a **method** `isPrime` that determines whether a number is a prime number. A number is **prime** if it is divisible only by one and itself. For example 11 is a prime number and 14 is not a prime number.

Write a `main` method to test your method.

A Sample output would be:

```
Enter a number: 3
3 is prime
```

Exercise 4-7 Character Count

Write a method named `count` that accepts two arguments: a `String` value, and a `char` value. Your method is to return the total number of times the second argument (the single character value) appears inside of the first argument. For example:

```
public class Count {
    public static void main(String args[]) {
        System.out.println( count("Hello", 'l') ); // displays 2
        System.out.println( count("Hello", 'o') ); // displays 1
        System.out.println( count("Hello", 'H') ); // displays 1
        System.out.println( count("Hello", 'h') ); // displays 0
    }
}
```

Exercise 4-8 Sum of Digits

Write a java method `sumOfDigits` that takes a string consisting of text and non-negative numbers as input and returns the sum of the digits of all the numbers in the string. For example for the string: "The year has 12 months, each month has 4 weeks and the week has 7 days.", your program should return 14, i.e. $1+2+4+7$

Exercise 4-9 Perfect Number To be discussed in the lab

A perfect number is a positive integer that is equal to the sum of its proper positive divisors. A proper divisor of an integer n is an integer between 1 (inclusive) and n (exclusive) that divides n with no remainder. For example, 6 is a perfect number because $6 = 1+2+3$.

Write an algorithm that prints all perfect integers that are less than or equal to a given integer n . The program should consist of four methods:

- A method that will calculate the sum of divisors of a given integer n
- A method that will check whether a number is a perfect number
- A method that will print all perfect numbers that are less than or equal a given integer n
- The main method

Exercise 4-10 Z-Algorithm - Midterm Spring 2013

Given a string S of length n , the **Z-Algorithm** produces a string Z where $Z.charAt(i)$ is the length of the longest substring starting from $S.charAt(i)$ which is also a prefix of S , i.e. the maximum k such that $S.charAt(j) == S.charAt(i+j)$ for all $0 < j < k$. Note that $Z.charAt(i)=0$ means that $S.charAt(0) != S.charAt(i)$.

The string x is a prefix of the string w if and only if $w = xy$.

Write a method that takes a string as argument and returns a string according to the **Z-Algorithm** presented above.

Assume that $Z.charAt(0)$ is always equal to 0.

For example:

```
zFunction("abab") -> 00301
zFunction("axbyaxba") -> 00003001
zFunction("ababababx") -> 006040200
zFunction("CSEN") -> 0000
```

Explanation of the first sample run: `zFunction("ababa") -> 00301`

For the first character a , the z value is 0. For the second character b , the Z value is 0 since any prefix of "ababa" starts with a . For the third character a , the longest substring starting from the third position is "aba" that is also a prefix of "ababa". Thus, the Z -value of a is 3. The fourth character b is having a Z -value 0. The last character a is having a Z -value 1.

Exercise 4-11 Persistent numbers - Midterm Spring 2015 To be discussed in the tutorial

- a) Write a method that takes a number and returns the multiplication of its digits.

- b) Persistent numbers are numbers where the sequential product of its digits eventually produces a single digit number. For example, take the number 764. $7 * 6 * 4 = 168$; $1 * 6 * 8 = 48$; $4 * 8 = 32$; and finally, $3 * 2 = 6$. The number of multiplication steps required to reach the single digit number from the given number is referred to as the persistence of the starting number. Thus the persistence of 764 is 4.

Write a method that takes an integer as input and returns its persistence.

Note: You have to use the method multiplyDigits described above with the most appropriate loop.

For example:

```
persistence(764) -> 4  
persistence(2) -> 0  
persistence(23) -> 1
```

Exercise 4-12 Run Length Decompression
 To be discussed in the tutorial

Given a String containing numbers and uppercase characters (A-Z), write a Java program that decompresses the String.

Example:

Input: 12W1B12W3B24W1B14W

Output: WWWWWWWWWWWBWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWWWW