Department of Computing and Software, McMaster University   Instructor: Dr. Reza Samavi
2XB3

Lab 03 Solutions for Practice problems

Topic   Java Tutorial - Functions

Additional information:
https://introcs.cs.princeton.edu/java/20functions/

- *Side effects.* A *pure function* is a function that, given the same arguments, always returns the same value, without producing any observable *side effects*, such as consuming input, producing output, or otherwise changing the state of the system. The function harmonic() is an example of a pure function. However, in computer programming, we often define functions whose *only* purpose is to produce side effects. Can you think about some examples of these types of functions?

Task 1: Write a Java program to print the n'th harmonic number.

```
/*********************************************************************
***********
 *  Compilation:  javac Harmonic.java
 *  Execution:    java Harmonic n
 *
 *  Prints the nth harmonic number: 1/1 + 1/2 + ... + 1/n.
 *
 *  % java Harmonic 10
 *  2.9289682539682538
 *
 *  % java Harmonic 10000
 *  9.787606036044348
 *

*********************************************************************
**********/

public class Harmonic {

    // returns 1/1 + 1/2 + 1/3 + ... + 1/n
    public static double harmonic(int n) {
        double sum = 0.0;
        for (int i = 1; i <= n; i++) {
            sum += 1.0 / i;
        }
        return sum;
    }

    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            int arg = Integer.parseInt(args[i]);
            double value = harmonic(arg);
            StdOut.println(value);
        }
    }

}
```

Task 2: Write a Java program to calculate the greatest common divisor

```
/*********************************************************************
***********
 *  Compilation:  javac Euclid.java
 *  Execution:    java Euclid p q
 *
 *  Reads two command-line arguments p and q and computes the 
greatest
 *  common divisor of p and q using Euclid's algorithm.
 *
 *  Remarks
 *  -----------
 *    - may return the negative of the gcd if p is negative
 *

*********************************************************************
**********/

public class Euclid {

    // recursive implementation
    public static int gcd(int p, int q) {
        if (q == 0) return p;
        else return gcd(q, p % q);
    }

    // non-recursive implementation
    public static int gcd2(int p, int q) {
        while (q != 0) {
            int temp = q;
            q = p % q;
            p = temp;
        }
        return p;
    }

    public static void main(String[] args) {
        int p = Integer.parseInt(args[0]);
        int q = Integer.parseInt(args[1]);
        int d  = gcd(p, q);
        int d2 = gcd2(p, q);
        StdOut.println("gcd(" + p + ", " + q + ") = " + d);
        StdOut.println("gcd(" + p + ", " + q + ") = " + d2);
    }
}
```

Task 3: Experiment that this Java program is spectacularly inefficient! Run the program for fibonacci(1), fibonacci(10) and fibonacci(100). Can observe the inefficiency? Open a task3.txt file and write what the function does to compute fibonacci(8) = 21. Describe what does it compute first and identifiy as formula how may time fibonacci(1) is being computed.

To see why it is futile to do so, consider what the function does to compute fibonacci(8) = 21. It first computes fibonacci(7) = 13 and fibonacci(6) = 8. To compute fibonacci(7), it recursively computes fibonacci(6) = 8 *again* and fibonacci(5) = 5. Things rapidly get worse. The number of times this program computes fibonacci(1) when computing fibonacci(n) is precisely F_n.

Task 4: Write a library of static method Hyperbolic.java that uses Math.exp() to implement the *hyperbolic* functions based on the definitions $\sinh(x)=(e^x-e^{-x})/2$ and $\cosh(x)= (e^x+e^{-x})/2$, with $\tanh(x)$, $\coth(x)$, $\mathrm{sech}(x)$, and $\mathrm{csch}(x)$ defined in a manner analogous to standard trigonometric functions.

```
/*********************************************************************
***********
 *  Compilation:  javac Hyperbolic.java
 *  Execution:    java Hyperbolic x
 *
 *  Static library of hyperbolic trigonometric functions.
 *
 *  Remark
 *  ------
 *  Java 1.5 includes more robust methods Math.sinh(),
 *  Math.cosh(), and Math.tanh().
 *

*********************************************************************
**********/

public class Hyperbolic {

    public static double cosh(double x) {
        return (Math.exp(x) + Math.exp(-x)) / 2.0;
    }

    public static double sinh(double x) {
        return (Math.exp(x) - Math.exp(-x)) / 2.0;
```

```java
    }

    public static double tanh(double x) {
        return sinh(x) / cosh(x);
    }


    public static void main(String[] args) {
        double x = Double.parseDouble(args[0]);
        StdOut.printf("sinh(%f) = %f\n", x, sinh(x));
        StdOut.printf("cosh(%f) = %f\n", x, cosh(x));
        StdOut.printf("tanh(%f) = %f\n", x, tanh(x));
    }
```

Practice Problems:

1. Write a static method `max3()` that takes three `int` arguments and returns the value of the largest one. Add an overloaded function that does the same thing with three `double` values.

```java
public static int max3(int a, int b, int c) {
        int max = a;
        if (b > max) max = b;
        if (c > max) max = c;
        return max;
    }

    public static double max3(double a, double b, double c) {
        double max = a;
        if (b > max) max = b;
        if (c > max) max = c;
        return max;
    }
```

2. Write a static method `majority()` that takes three `boolean` arguments and returns `true` if at least two of the arguments are `true`, and `false` otherwise. Do not use an `if` statement.

```java
public static boolean majority(boolean a, boolean b, boolean c)
{
        return (a && b) || (a && c) || (b && c);
    }
```

3. Consider the following recursive function.

```java
public static int mystery(int a, int b) {
   if (b == 0)     return 0;
   if (b % 2 == 0) return mystery(a+a, b/2);
   return mystery(a+a, b/2) + a;
}
```

What are the values of mystery(2, 25) and mystery(3, 11)? Given positive integers a and b, describe what value mystery(a, b) computes.

Solution: 50 and 33. It computes a*b. If you replace + with *, it computes a^b.