Prof. Dr.-Ing. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 07

Submission of solutions until 3:00 p.m. at 22.12.2021
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way!
- Please comment your solutions, so that we can easy understand your ideas!
- If you have questions about programming or the homeworks, just ask you teachers!

Exercise 1 (Evaluation: Predefined `main` Method)

This task deals with two-dimensional (**`double`**) arrays and how they can be utilised as matrices. Their task is to implement methods that make it possible to multiply two matrices together. In case you have lost track of how matrix multiplication is performed, you can find guides under following links[1][2].

For this task you have to implement the following methods:

```
boolean isValid(double[][] matrix1, double[][] matrix2)
double[][] mulMatrices(double[][] matrix1, double[][] matrix2)
```

The method `isValid()` is used to determine if a matrix multiplication is possible. Remember, that you can only multyply two matrices if the number of collumns of `matrix1` is equal to the number of rows of `matrix2`.

The method `mulMatrices()` is used to implement the actual matrix multiplication. It returns a two-dimensional **`double`** array, containing the result of the multiplication. In case that both matrices cannot be multiplied because `isValid()` returns **`false`**, simply return the **`null`** reference

A suitable `main` method for testing your two implementations is already predefined there.

[1] `https://www.mathsisfun.com/algebra/matrix-multiplying.html`
[2] `https://en.wikipedia.org/wiki/Matrix_multiplication`

Exercise 2 (Evaluation: predefined `main` method)

Consider the recursively defined sequence $(f_n)_{n=0,1,...}$ of integers f_n:

$$\begin{aligned} f_0 &:= 0 \\ f_1 &:= 2 \\ f_2 &:= 4 \\ \text{and in case } n \geq 3: \quad f_n &:= (f_{n-1} + f_{n-2}) \cdot f_{n-3} \end{aligned}$$

1. Implement a "`static double frec(int n)`" method to calculate the n" th value f_n of this sequence in the form of a *recursive algorithm*.
2. Implement a "`static double fdyn(int n)`" method to calculate f_n using *dynamic programming*[3].

You should write both partial solutions into the given file. A suitable `main` method for testing your two implementations is already predefined there.

Exercise 3 (Evaluation: predefined `main` method)

Familiarize yourself with the methods of the `String` class[4]:

1. Create a method

   `static int countOccurrence(String haystack, String needle)`

   that counts at how many positions the string `needle` occurs as a substring in `haystack` For example, the call `countOccurrence("abbbba", "bb")` should return the value 3.

2. Create another method

   `static String extractTag(String s)`

   with the following property: For an argument `s` of the form

   prefix '`[`' *infix* '`]`' *suffix*

   the function `extractTag` should return the substring *Infix*. You can assume that '`[`' and '`]`' both occur exactly once in the correct order in `s`.

   At `extractTag("1234[5678]9")` the string `5678` should be found and returned.

[3] See the slides of chapter 3 part 4

[4] `http://docs.oracle.com/javase/8/docs/api/java/lang/String.html`

Exercise 4 (PCP - Post correspondence problem)

Create a method

```
static int pcpTest(String[] x, String[] y, int[] t)
```

with two `String` arrays (`x` and `y`) of the same size and one **`int`** array (usually of a different size) as parameter. It is to be tested whether `t` is a solution to the post correspondence problem (PCP) [5] of `x` and `y` is.

This test is to proceed as follows:

- First, the method should test the validity of the parameters: It should return the value `-1` if one of the following cases occurs:
  - if `x`, `y` or `t` is a **`null`** reference,
  - if the arrays `x` and `y` have different numbers of elements,
  - if one of the arrays `x` or `y` contains a **`null`** reference instead of a string,
  - if one of the strings in one of the arrays `x` or `y` is the empty string `""`,
  - if the array `t` has the length 0,
  - if one of the `t.length`-many values in the field `t` is negative or $\geq$ `x.length` is.

  These cases are invalid parameters for the post correspondence problem.
- If the parameters are valid, the following must be tested: If the string `testX` resulting from the concatenation of the strings

  `x[t[0]] + x[t[1]] + ... + x[t[t.length-1]]`

  matches the string `testY` from the concatenation of

  `y[t[0]] + y[t[1]] + ... + y[t[t.length-1]]`

  If yes, `1` should be returned; if no, `0` should be returned.

Example: With `x={"aba", "ba", "b"}` and `y={"a", "ba", "bab"}` there is a match for `t={0,1,1,2}`:

```
  t:     0      1      1      2
testX= aba  +  ba  +  ba  +   b   =abababab
testY=  a   +  ba  +  ba  +  bab  =abababab
```

[5] see `https://en.wikipedia.org/wiki/Post_correspondence_problem`