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

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 06

Submission of solutions until 3:00 p.m. at 15.12.2021
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: Numbers)

Add the following five methods to the predefined program:

1. `sub`: three `int` parameters, `int` return value;
   the smallest number is subtracted from the largest number.
2. `mul`: three `int` parameters, `int` return value;
   the two largest numbers are multiplied by each other.
3. `mean`: thre `int` parameters, `double` return value;
   the arithmetic mean $\frac{a+b+c}{3}$ of the three parameters is determined as `double` and returned. Example: $\frac{1+2+2}{3} = 1,666...$
4. `allEqual`: three `int` parameter, `boolean` return value;
   the `boolean` value `true` is returned if all three parameters have the same value, otherwise the value `false`.
5. `prime`: one `int` parameter, `boolean` return value;
   return `true`, if the `int` number is positive and a prime numbers. Otherwise return `false`

You may only change the default `main` method as follows: If you have implemented a subtask, remove the comment characters from the line beginning of the corresponding case of the `switch` statement. This will activate the subtask.

The `main` method first reads in up to four numbers `st`, `a`, `b`, `c`. The first number `st` determines the subtask to be considered. An example can be found on the next page.

Watch out: The methods to be written by you must all be provided with the modifier `static`!

```
Subtask: 3
Test Values: 1 2 2
1.6666666666666667
```

Exercise 2 (Evaluation: Text)

Write two static methods to determine the area $A(s) = s^2$ of a square and the volume $V(s) = A(s) \cdot s$ of a cube with side length s:

```
double squareArea(double side)
double cubeVolume(double side)
```

During implementation, the function for the volume should call the function for the area.

Then write a `main` method, which in turn calls `cubeVolume` appropriately. Your program should read two `double` numbers a, b with `sc.nextDouble()`, where you can assume that $0 \leq a < b$ applies. Then a small table of 11 values each for area and volume of the cube shall be output, where the page length s results from $s = a + n \cdot \frac{b-a}{10}$ for $n = 0, 1, 2, ..., 9, 10$.

To output the `double` numbers, use the `System.out.format` method. For example, `System.out.format("%8.2f", x)` outputs a `double` number `x` with 8 characters width with 2 decimal places. This allows you to reproduce exactly tables of the following form:

```
Range: 0 5

|    Side   |    Area    |   Volume   |
|-----------|------------|------------|
|     0.00  |      0.00  |      0.00  |
|     0.50  |      0.25  |      0.13  |
|     1.00  |      1.00  |      1.00  |
|     1.50  |      2.25  |      3.38  |
|     2.00  |      4.00  |      8.00  |
|     2.50  |      6.25  |     15.63  |
|     3.00  |      9.00  |     27.00  |
|     3.50  |     12.25  |     42.88  |
|     4.00  |     16.00  |     64.00  |
|     4.50  |     20.25  |     91.13  |
|     5.00  |     25.00  |    125.00  |
```

Exercise 3 (Evaluation: Numbers)

Normally a year is a leap year when the year number is dividable by 4 without remainder. This does not apply if it is dividable by 100 without a remainder. However, if the number is dividable by 400 even without remainder, it is still a leap year.

Implement to methods:

- `isLeapYear` checks if the input parameter is a leap year, corresponding to the previous mentioned rules.
- `previousLeapYear` returns the leap year that occurred before the entered year.

`previousLeapYear` should call the method `isLeapYear` (possibly several times). Decide for each method which data to pass and what the return type is.

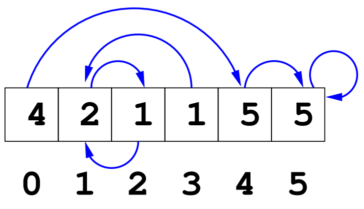
Write a `main` function that allows corresponding tests of `previousLeapYear`, suitably for the following example:

```
Year:  2018
Previous Leap Year: 2016
```

Exercise 4 (Evaluation: Numbers)

Consider a field `array` analogous to the scavenger hunt examples (i.e., it's about repeatedly setting `i=field[i]`). However, all entries in the field are certainly so small that they can be used as an index for the field again. Every 'scavenger hunt' must therefore run in a loop at some point:

With the values `4 2 1 1 5 5`, for example, the following transitions occur in the array and thus obviously also two loops (loop 1 is 5-5-5-..., loop 2 is 1-2-1-2-...):



Now determine how many loops there are in an input field (including the indexes to the indexes leading to the loops).

In the example, the indices 0,4,5 lead into loop 1, the indices 1,2,3 lead into loop 2:

```
Size:  6
Content: 4 2 1 1 5 5
Number of Loops:  2
Assignment of Indexes: 1 2 2 2 1 1
```

Advice: Use a second field `marked` of type `int`, in which you enter for each index the loop in which it is located, if this is already known. So meaningful initial values for the `marked` field are `0`.

You can now proceed as follows:

1. At the beginning of the algorithm no loop is known, so initialize a counter k with the value 0.

2. Then look (repeatedly, see below) at the smallest index `i` that has not yet been assigned to a loop, where `marked[i]` still has the value `0`.

   (At the beginning there is always `i=0`*. This procedure also results in a unique numbering of the loops as "loop 1", "loop 2", etc.).*

3. Then check (e.g. as in task 4 of Assignment 4) whether you reach an already known loop from `i` (i.e. an index `j` is reached from `i`, where `marked[j]` has a value >`0`).

4. If a known loop is reached, save the value found in the loop `marked[j]` in a variable `s`.

5. But if no known loop is reached, you have found a new loop; so increment `k` and also set `s` to the new value of `k`.

6. Now set the `marked` field to the value `s` for `i` and for all indexes that can be reached from `i`.

7. As long as the `marked` field still contains `0` values, repeat this procedure from step 2.

8. At the end, output the value of k and the values of the `marked` field.

It is best to first execute the algorithm "by hand" on some of the evaluation examples before implementing it!