Scripting Languages – Tasks for the 2nd Cycle of Laboratory Exercises

April 2024

1 Introduction

The second cycle of lab exercises covers the basics of the **Perl programming language**, both theoretically and through practical application.
Students are required to prepare for the lab exercises by independently solving a set of simple tasks.
For tasks involving file operations (content search, renaming, etc.), students must create appropriate test files to demonstrate script functionality.
A prerequisite for attending the lab exercise is submitting (uploading) solution files via the Ferko system: https://ferko.fer.hr/ferko/.
Submission must follow the instructions regarding **file naming and script execution (including parameter usage, if applicable).**

During the lab session, students will:

- Take a **short quiz**,
- **Solve one programming task** independently on the computer,
- **Defend their preparation tasks** in front of an assistant.

1.1 Lab Execution Protocol

In short, the lab session protocol is as follows:

1. Students must complete the assigned tasks and upload their solution files to Ferko.
2. Students must attend the lab session at their scheduled time.
3. At the beginning of the lab, students take a short **quiz** (multiple choice; correct answer = 0.5 pts, incorrect = -0.125 pts).
4. During the lab, students are given a **programming task** (exit test) to solve on-site and submit via Ferko. **Be sure to finalize (lock) your submission!**
5. During the lab, students must **present their pre-solved tasks** to the assistant, who evaluates them.

1.2 Lab Resources

The exercises are conducted in faculty labs. The computers have **Cygwin** installed with necessary tools (**bash, perl, etc.**).

Task 1

Write a Perl script that will **ask the user to input** a character string and a number (n).
The script should print the entered string **n times**, each on a **new line**.

Upload instruction: Name the script `zadatak1.pl`.

Task 2

Write a Perl script that reads a **list of numbers** into an array, calculates and prints their **arithmetic mean**.

Upload instruction: Name the script `zadatak2.pl`.

Task 3

There are two **web server log files** available in the course repository. These are text files, generated daily, with the date embedded in their names.
Each line represents one server access.

Write a **Perl script** that prints the number of accesses **per hour for each day**.
The script accepts log file names as **command-line arguments** (they may not be in the current directory).
If no files are specified, the script should read from **standard input** (using <>).

Sample output:

```
$ ./zadatak3.pl ../TestExamples/localhost_access_log*.txt
Date: 2008-02-24
hour : access count
------------------------------
00 : 99
01 : 62
02 : 15
...
Date: 2008-02-25
hour : access count
------------------------------
00 : 378
01 : 68
...
```

Upload instruction: Name the script `zadatak3.pl`.

Task 4

A file contains lab attendance records and submission times of students' exit tests, formatted as:

```
JMBAG;LastName;FirstName;TimeSlot;Submitted
0036438919;Bagarić;Magdalena;2011-03-14 08:00 11:00 A209;2011-03-14
08:45:02
...
```

Write a **Perl script** that checks for each student whether they submitted their test **within the first hour** of their lab session (assuming sessions start on the hour).
Print students who **did not** meet this requirement.
If no filename is provided, the script should read from **standard input**.

Sample output:

```
$ ./zadatak4.pl ../TestExamples/lab-closed.csv
0036436684 Lombarović Mladen - PROBLEM: 2011-03-14 11:00 --> 2011-03-14
12:08:26
...
```

Upload instruction: Name the script `zadatak4.pl`.

Task 5

Write a Perl script that reads **student scores** from a file provided as a **command-line argument**.
The first line contains weight **factors** for each score component.
Subsequent lines contain: student ID, name (as one field), and scores for each component.
Fields are **semicolon-separated**; missing scores are marked with `-`.
The file may contain **comments** (lines starting with `#`) and **blank lines**—these should be ignored.

Example input:

```
# component weights
0.15;0.20;0.30;0.10;0.10;0.10;0.05

# student results
0036438919;Bagarić;Magdalena;91.5;90.75;88.25;100;87.5;87.5;90
...
```

The script should **calculate total scores**, sort students by total (descending), and print a **ranked list** with name, ID, and score.

Sample output:

```
$ ./zadatak5.pl ../TestExamples/results.csv
Ranking list:
```

```
--------------------
1. Bagarić, Magdalena (0036438919) : 90.35
2. Pavlinović, Matija (0036427706) : 82.20
...
```

Upload instruction: Name the script `zadatak5.pl`.

Task 6

Write a Perl script that **counts words** with a **common prefix** in the provided file(s).
The **prefix length** is the **last argument**, preceded by one or more **filenames**.
If no file is specified, the script reads from **standard input**.

Words are separated by spaces, tabs, newlines, and punctuation.
Case-insensitive comparison is required.
Sort prefixes alphabetically and print their occurrence count.

Sample file (notturno.txt):

```
Noćas se moje čelo žari,
noćas se moje vjede pote;
...
```

Sample output:

```
$ ./zadatak6.pl ../TestExamples/notturno.txt 4
čelo : 1
dubi : 1
duša : 1
...
```

To support Croatian characters (UTF-8), use:

```
use open ':locale';
use locale;
use utf8;
```

Shell configuration:

```
export LC_ALL=hr_HR.UTF-8
```

Upload instruction: Name the script `zadatak6.pl`.