01UDFOV/01TXYOV – WEB APPLICATIONS I

GETTING STARTED WITH NODE.JS

During this first lab, you will start to become acquainted with JavaScript (in Node.js) by putting in practice what you saw in the lectures of the first two weeks. In addition, this lab will put the basis for the following ones, by kickstarting a project that will continue up to the end of the course.

EXERCISE 0 - PREPARATION

First of all, check that your *Node.js* installation is working within Visual Studio Code.

Create a new project that, for each string in an array, returns a new string made of the first two and the last two characters of the original string. The new string should replace the old one in the same array.

e.g., '*spring*' yields '*spng*'

If the word is shorter than two characters, return the empty string.

EXERCISE 1 – A TODO MANAGER

Given a series of tasks (i.e., actions that the user wants to do in the future) implement a `todo_manager` program.

In particular, a task is made of the following fields:

- a textual *description* (mandatory)
- whether it is *urgent* or not (default to "not urgent")
- whether it is a *private* or a *shared* task (default to "private")
- a *deadline* (i.e., a date with or without a time, optional)

The program should perform 4 actions:

1. insert a new task;
2. remove a task;
3. show all existing tasks, in alphabetic order;
4. close the program.

At startup, the program shows a menu with the 4 options and, for each choice, performs the requested action. After each action (except for action 4), the program returns to the prompt for actions.

To insert a task, the program will ask for the necessary fields, one per line.

To remove a task, the user should indicate the exact textual description of the task to be removed.

Hints:

1. *To read from the terminal, you can use the* `readline-sync` *module:* *https://www.npmjs.com/package/readline-sync*.
   To install the module, fire up a terminal, go in the project folder and type "`npm install readline-sync`*" (you will need an Internet connection).*
2. *You have 3 options to execute the program:*
   a. *manually, in a terminal console (the one integrated in Visual Studio Code, or the system console);*
   b. *in Visual Studio Code, to read from the terminal, you need to create a launch configuration (launch.json) from the Run activity (in the "Activity Bar" on the left) and add a new configuration:*
      `"console": "integratedTerminal"`
   c. *The Visual Studio Code configuration suggested above is already set in the project available on GitHub, that can be cloned at https://github.com/polito-WA1-2020/lab1-node (which will also host the solution of the lab).*
3. *To create and handle dates, you can exploit the* `Date` *object.*

EXERCISE 2 – EXTENDING TASK DELETION...

Modify the program developed in the previous exercise to find and remove all the tasks with a given deadline.

For example, when user types "2020-03-15", the program will use the provided information to delete all tasks whose deadline is on the specified date.

EXERCISE 3 – ... AND GETTING RID OF THE PAST

Extend the program developed in the previous exercises to automatically delete a task when it expires. You can test it by adding tasks whose date is today (e.g., "2020-03-20") and with a time corresponding to a few minutes after the task insertion.

Extra: beware, 2020-03-17 should come later than 2020-03-17 14:21!

Hint: Use the Node.js' `setTimeout()` *function, and the* `Date` *object to compute the timeout value when the task is inserted. Please, be aware that* `setTimeout()` *does not play well with loops, consider replacing the loop with* `setInterval()`.