Task 1: Understanding the Architecture of the WWW and the HTTP protocol

The architecture of the world wide web is one of the most important used patterns in the internet. Have a look at the following materials and make yourself familiar with the architecture behind WWW:

- Design Principles and Architecture:
  - https://www.w3.org/DesignIssues/Principles.html
  - https://www.w3.org/DesignIssues/Architecture.html

HTTP is a popular communication protocol between terminal devices in the web. Read the following materials and understand the history and concepts behind HTTP:

- https://hpbn.co/brief-history-of-http/
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
- https://www.lehigh.edu/~inwww/old_seminar/url-basics.html

(Optional) If you are interested to learn more about the web, you can have a look at the following resources:

- https://www.w3.org/History/1989/proposal.html
- https://tools.ietf.org/html/rfc2616

Task 2: Implementing a Webserver

In this task, you are required to write a webserver from scratch. You can use Java, C#, or C/C++. The webserver should meet the following requirements. It should:

- be a command line program
- use standard libraries
- be possible to configure the port (either in the source code or as argument at start-up)
- respond to HTTP GET requests, specifically
  - the request "/" (root) should respond with "Hello World"
  - the request "/s" should an HTML-response stating the IP address of the client
- (Optional) be possible to block IP addresses from receiving the response, such as requests from a given IP address will always receive an Error 404 (file not found)
- (Optional) it should be possible to block IP addresses from receiving the response, e.g. requests from a given IP address will always receive an Error 404 (file not found)
- (Optional) change the response for "/" each time it is called again from the same IP address

You can use the following references for the base implementation:

- http://www.onjava.com/pub/a/onjava/2003/04/23/java_webserver.html (Java)
- https://msdn.microsoft.com/en-us/library/system.net.httplistener(v=vs.110).aspx (C#)

Furthermore, you are welcome to build on examples of programs you find in the internet (preferably open source licenses). Please make sure to reference all resources, examples, and information that you use to write your program. Please include these references in the source code of your program as comments.

To hand in: source file of your program with comments and the password
To demonstrate: your working program

Task 3: Crawling the Web: Write a Web-Client and Parsing a Response

Write an automated Web Client in a programming language of your choice (e.g. Java, C#, Javascript, Python, alternatively you can use Scrapy) that downloads the web page from https://ubicomp.net/sw/task1.php and searches in this web page for the following elements: A URL and a password. This information is only available for a few minutes per day, hence you need to automate the task. This fictional password would allow you to access the webpage. Please note, that the password is only available at a random time each day for a few minutes.

You are welcome to build on examples of programs you find in the internet (preferably open source licenses). Please make sure to reference all resources, examples, and information that you use to write your program. Please include these references in the source code of your program as comments.

To hand in: source file of your program with comments and the password
To demonstrate: your working program

Interested in more? Build your own web crawler with Scrapy:
https://scrapy.org/
https://www.makeuseof.com/tag/build-basic-web-crawler-pull-information-website-2/

Task 4: Node.JS webserver Tutorials

Get familiar with NodeJS and npm. You can search the web for tutorials or use one of the following links:

- https://www.w3schools.com/nodejs/default.asp
- https://www.tutorialspoint.com/nodejs/index.htm
- https://www.youtube.com/watch?v=U8XF6AFGqlc
- https://www.youtube.com/watch?v=TlB_eWDSMt4

We will use NodeJS in the next exercises.

Task 5: Web Server on ESP8266 (Optional, Alternative for Task 2)

Program a basic Web-Server on the ESP8266 that allows to control an LED by an HTTP-request and allows to read out the analog value of a pin using a HTTP request.

You can find the firmware and some tutorials here:
Firmware:

- https://nodemcu-build.com/ (build your own firmware)
- http://thomaskosch.com/wp-content/files/nodemcu_firmware.bin (prebuilt firmware)

IDE:

- http://esp8266.ru/esplorer-latest/?f=ESPlorer.zip (Lua IDE)
- https://www.arduino.cc/en/Main/Software (Arduino IDE)

Flash tools:

- https://github.com/nodemcu/nodemcu-flasher (flash nodemcu firmware, windows only)
- https://github.com/espressif/esptool (flash nodemcu firmware, all platforms, requires python)

Tutorials:

- https://roboindia.com/tutorials/starting-with-lua-on-esp8266-wifi-module (Lua tutorial)
- https://roboindia.com/tutorials/esp8266-home-automation-lua (Lua tutorial)
- https://randomnerdtutorials.com/esp8266-web-server/ (Arduino tutorial)

Task 6: Crawling the Web 2.0 (Optional, Alternative for Task 3)

Implement a program (Java, C#, Javascript) which reads a file with a list of URLs to monitor (e.g. newspages). Your program should read two command line arguments:

- A keyword which is used to be searched on the webpages
- An argument which specified how often the webpages should be checked (e.g. entering a „10" can be interpreted for checking the webpages every 10 seconds)

Think about how to log and represent your results. For example, you could log the keyword together with the webpage and the time the keyword was found in a separate log file or display it on a webinterface.

Modalities for handing in your results:

Please hand in a zip file of your source code (**Deadline: 29.04.18, 23:59**). Upload a zip file to Uniworx with the following format: Studentnumber_Lastname_Firstname (German: Matrikelnummer_Nachname_Vorname). The zip file should contain the following file structure:

- Studentnumber_Lastname_Firstname
  - Task 2
    - Commented and referenced sourcecode
  - Task 3
    - Commented and referenced sourcecode + the password

For example, a student with the name Joe Doe has the student number 123456. The file would look this way: “123456_Joe_Doe.zip”.