**HTTP protocol and HTML (5p)**

**Part I. HTTP protocol (1p)**

1. Run Google Chrome and go to some website (everyone's sites should be different!). Run the debugger (F12), go to the Network tab in it and refresh the page. Select some resource loaded by the browser using the GET method. To see it in its original form (RAW) and insert it into the report, you will have to work hard. Open the website <https://curlconverter.com/> in the new browser tab. Go back to the previous tab with the opened debugger, right-click in the left column on the resource you selected, select "Copy" → "Copy as cURL". Paste the copied command on the website <https://curlconverter.com/> and go to the HTTP tab. Copy the RAW request to the report.
To obtain the RAW response we need the web-service <https://reqbin.com/curl>. Paste the copied command there, **Run** it and copy the **RAW** browser response to the report.
Indicate next to each header of the browser request and of the server response what it means (you should find it in google, use query: «http header ....»).
2. Intercept the browser's request to server using the **GET method with parameters**.
Parameters come after the"? " sign. Write this request in the report in RAW format. Specify the name of the CGI program running on the server, the names of the parameters passed to it, and the values of these parameters.
3. Intercept browser request to some servers using the **POST method** with data format application/x-www-form-urlencoded. Write it (along with the body) in RAW format in the report. Specify the name of the CGI program running on the server, the names of the parameters passed to it, and the values of these parameters. Specify which parameters are passed to the CGI application via the QUIERY\_STRING environment variable, and which are passed via the standard input stream.
4. Intercept the browser's request with cookie. Write this request in the report in RAW format. How long will this cookie be stored?
5. Intercept the server response that sets a Cookie for the browser. Write it in the report in RAW format. How long will you store this Cookie? Which domain is this cookie valid for?

**Part II. HTML and CSS**

**(4p + 3p bonus in case of a beautiful website)**

Create a quick website about your hobbies with photos that would work normally on large-format monitors and mobile phones. It should consist of several (>=2) beautifully designed html pages linked with hyperlinks. For quick work, use ready-made templates on bootstrap: <https://bootstrapmade.com/free-website-templates/>

For students with experience in web development, Part II is optional. Instead, you can show your own web project.

**Installing and configuring a web server (3p)**

Using shared hosting in the Nginx web server, implement the operation of two web sites inside the same docker container, corresponding to two different domain names (let's call them site1.ru and site2.ru).

1. Create and run the Nginx docker container (use the official image!).

2. Copy the Nginx settings folder to the current directory

 docker cp containerId:/etc/nginx etc\_nginx

then recreate the container by mounting the etc\_nginx folder to the container's /etc/nginx.

3. At the end of the etc\_nginx/conf.d/default.conf configuration file, add the minimum configurations for the two sample sites:

server {

 server\_name domainNameSite1;

 root pathToTheSiteInsideContainer.

}

4. Copy some html pages to the folders with site content inside the docker container.

5. Ask Nginx to update the configuration by restarting the container (docker container restart containerId)

7. On your personal computer, edit the /etc/hosts file (sudo mcedit /etc/hosts command) so that the domain names of your sites correspond to the IP address of the docker container (you can view it with the command

docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' containerId

8. Check working of sites in the browser. If something doesn't work, check the container's log.