Задание 00

1. Разработайте приложение (сервер) **06-00** предназначенное для обработки следующих запросов.

HTTP-метод	URI	Задание
GET	/connection?set=***set***	01
GET	/headers	02
GET	/parameter?x=***x***&y=***y***	03
GET	/parameter/***x***/***y***	04
GET	/socket	05
GET	/resp-status?code=***c***&mess=***m***	06
POST	/formparameter	07
POST	/json	08
POST	/xml	09
GET	/files	10
GET	/files/***filename***	11
GET/POST	/upload	12

Задание 01 /connection?set=***set***

2. При GET-запросе **/connection** в окно браузера вывести текущее значение параметра **KeepAliveTimeout.**
3. При GET-запросе **/connection?set=*set*** установить новое значение системного параметра **KeepAliveTimeout = *set*** и вывести в окно браузера сообщение, что установлено новое значение параметра **KepAliveTimeout=*set***.
4. Продемонстрируйте влияние системного параметра **KeepAliveTimeout** на работу приложения.

Задание 02 /headers

5. Отобразите в окне браузера все заголовки запроса и ответа.
6. Объясните назначение каждого заголовка.

Задание 03 /parameter?x=***x***&y=***y***

7. Проанализируйте значения параметров **x** и **y.**

8. Если **x** и **y** имеют числовые значения, то выведите в окно браузера сумму, разность, произведение и частное этих чисел
9. Иначе выведите сообщение об ошибке.

Задание 04 /parameter/***x***/***y***

10. Проанализируйте значения параметров **x** и **y**.
11. Если **x** и **y** имеют числовые значения, то выведите в окно браузера сумму, разность, произведение и частное этих чисел.
12. Иначе выведите **URI**.

Задание 05 /socket

13. При получении этого запроса, в окно браузера выведите ip-адрес, порт клиента и ip-адрес и порт сервера.

Задание 06 resp-status?code=***c***&mess=***m***

14. При получении этого запроса, сформируйте ответ, имеющий статус, заданный значением ***c***, и пояснение к статусу, заданное значением ***m***.

Задание 07 /formparameter

15. Используйте HTML-форму, включающую теги **input** с **type**: **text**, **number**, **date**, **checkbox**, **radiobutton**, тег **textarea**, а также два тега **input type=submit**, имеющих одно и тоже имя, но разные значения.
16. В окно браузера выведите значения параметров, полученных в запросе.

Задание 08 /json

17. Принимайте POST-запросы, содержащие данные в json-формате и отправляйте ответы в json-формате.
18. Сообщение в запросе имеет следующую структуру:

```
{
    "__comment": " Запрос.Лабораторная работа 8/10",
    "x": 1,
    "y": 2,
    "s": "Сообщение",
    "m":["a","b","c","d"],
    "o":{"surname":"Иванов", "name":"Иван"}
}
```

19. Сообщение в ответе имеет следующую структуру:

```
{
    "__comment": " Ответ.Лабораторная работа 8/10",
    "x_plus_y": 3,
    "Concatination_s_o": "Сооощение: Иванов, Иван",
    "Length_m": 4
}
```

Поле **x+y** ответа содержит сумму полей **x** и **y** запроса.
Поле **Concatination_s_o** ответа - конкатенацию полей **s** и свойств объекта **o** запроса.
Поле **Length_m** ответа - количество элементов в массиве **m** запроса.

20. Проверьте работоспособность приложения с помощью POSTMAN.

Задание 09 /xml

21. Принимайте POST-запросы, содержащие данные в xml-формате и отправляйте ответы в xml-формате.
22. Сообщение в запросе имеет следующую структуру.

```
<request id = "28">
    <x value = "1"/>
    <x value = "2"/>
    <m value = "a"/>
    <m value  = "b"/>
    <m value  = "c"/>
</request>
```

23. Количество элементов **x** и **m** в запросе может быть произвольным.

24. Сообщение в ответе имеет следующую структуру:

```
<response id ="33" request="28">              <!-- ответ на запрос  28 -->
   <sum    element = "x" result="3" />       <!-- сумма значений элементов x  -->
   <concat element = "m" result = "abc" />   <!-- конкатенайия всех элементов  m  -->
</request>
```

25. Элемент **sum** в ответе один и содержит сумму всех значений элементов **x** (в атрибуте **result**).
26. Элемент **concat** в ответе один и содержит сумму всех значений элементов **m** (в атрибуте **result**).
27. Проверьте работоспособность приложения с помощью POSTMAN.

Задание 10 /files

28. В ответ на запрос высылается ответ с заголовком **X-static-files-count: *n***, где ***n*** - количество файлов в директории **static**. Используйте функции модуля **fs**.
29. Проверьте работоспособность приложения с помощью POSTMAN.

Задание 11 /files/***filename***

30. В ответ на запрос высылается ответ, пересылающий файл с именем ***filename*** из директории **static**.
31. Если файл ***filename*** не найден возвращается ответ со статусом 404.
32. Проверьте работоспособность приложения с помощью браузера.

Задание 12 /upload

33. В ответ на GET-запрос к **/upload** высылается web-форма, позволяющая отправить POST-запрос к **/upload**, присылающий серверу файл.
34. Сервер сохраняет файл в директории **static**.
35. Проверьте работоспособность приложения с помощью браузера.

Задание 13 Ответьте на следующие вопросы

36. Поясните назначение заголовка **Content-Type**.

37. Поясните назначение заголовка **Accept.**
38. Для чего используется значение **multipart/form-data** заголовка **Content-Type.**
39. Как с помощью тега **form,** обеспечить значение **multipart/form-data** заголовка **Content-Type.**
40. Какое значение заголовка **Content-Type** отправляется тегом **form** в запросе по умолчанию.
41. Где и в каком формате передаются параметры в GET-запросе?
42. Где и в каком формате передаются параметры в POST-запросе?
43. Поясните понятие **JSON**?
44. Поясните понятие **XML**?