ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет «Инфокоммуникационных технологий»
Направление подготовки «09.03.03 Мобильные и сетевые технологии»

О Т Ч Е Т

Тема задания: **Лабораторная работа №1**

Выполнил:
Студент Шоломов Д. О. К33402
(Фамилия И.О.) номер группы

Проверил:
Преподаватель Говоров. А. И.
(Фамилия И.О)

Задание 1

Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

```python
import socket


def task1():
    sock = socket.socket()
    sock.bind(('', 11111))
    sock.listen(1)
    conn, _ = sock.accept()
    while True:
        data = conn.recv(1024)
        if not data:
            break
        print(data.decode('utf-8'))
        conn.send(bytes('Hello, client', "utf-8"))
    conn.close()


if __name__ == '__main__':
    task1()
```

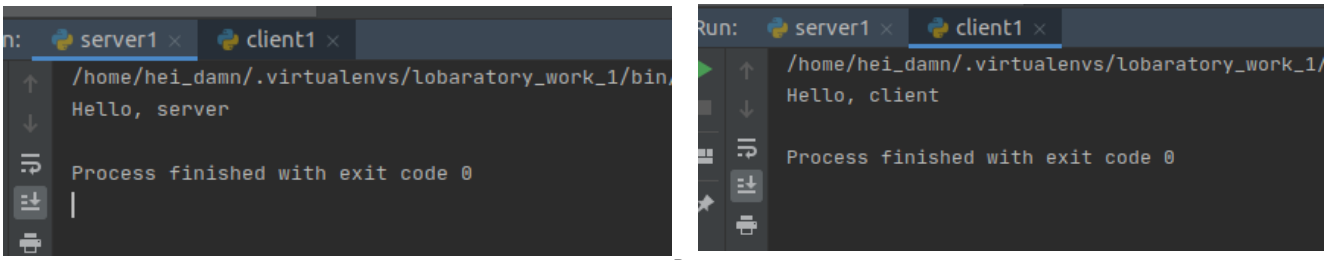
Серверная часть

```python
import socket


def task1():
    sock = socket.socket()
    sock.connect(('localhost', 11111))
    sock.send(bytes('Hello, server', "utf-8"))
    data = sock.recv(1024)
    sock.close()
    print(data.decode('utf-8'))


if __name__ == '__main__':
    task1()
```

Клиентская часть



Результат выполнения программы

Задание 2

Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту. Вариант задания: площадь параллелограмма.

Серверная часть

```python
import socket


def task2():
    sock = socket.socket()
    sock.bind(('', 22222))
    sock.listen(1)
    conn, _ = sock.accept()
    while True:
        data = conn.recv(1024)
        if not data:
            break
        inp = data.decode('utf-8').split()
        if len(inp) != 2:
            conn.send(bytes('Неверный формат ввода', "utf-8"))
            break
        conn.send(bytes(str(int(inp[0])*int(inp[1])), "utf-8"))
    conn.close()


if __name__ == '__main__':
    task2()
```

Клиентская часть

```python
import socket


def task2():
    sock = socket.socket()
    sock.connect(('localhost', 22222))
    print("Введите высоту и сторону параллелограмма через пробел:")
    sock.send(bytes(input(), "utf-8"))
    data = sock.recv(1024)
    sock.close()
    print(data.decode('utf-8'))


if __name__ == '__main__':
    task2()
```

Результат выполнения программы

```
/home/hei_damn/.virtualenvs/lobaratory_work_1/bin/python "/
Введите высоту и сторону параллелограмма через пробел:
10 5
50

Process finished with exit code 0
```

Задание 3

Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

```python
import socket


def task3():
    sock = socket.socket()
    sock.bind(('', 33333))
    sock.listen(1)
    conn, _ = sock.accept()
    while True:
        data = conn.recv(1024)
        if not data:
            break
        with open('index.html', 'r') as file:
            html = file.read()
        conn.sendall(bytes(f'HTTP/1.0 200 OK\nContent-Type: text/html\n\n{html}', 'utf-8'))
    conn.close()


if __name__ == '__main__':
    task3()
```

Серверная часть

```python
import socket


def task3():
    sock = socket.socket()
    sock.connect(('localhost', 33333))
    sock.send(bytes('.', "utf-8"))
    data = sock.recv(1024)
    sock.close()
    print(data.decode('utf-8'))


if __name__ == '__main__':
    task3()
```

Клиентская часть

Результат выполнения программы

```
client3 × server3 ×
/home/hei_damn/.virtualenvs/lobarator
HTTP/1.0 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>

Process finished with exit code 0
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
```

Задание 4

Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов.
Вариант: многопользовательский чат

Серверная часть

```
from threading import Thread
import socket


def connect():
    while True:
        conn, addr = sock.accept()
        users.append([conn,  addr])
        print(addr)
        th_listen = Thread(target=resend, args=(conn,))
        th_listen.start()


def resend(conn):
    while True:
        try:
            data = conn.recv(1024)
            message = str(data.decode("utf-8"))
            for user in users:
                if user[0] != conn:
                    user[0].send(message.encode())
        except Exception:
            for user in users:
                if user["connection"] == conn:
                    users.remove(user)
            break


if __name__ == "__main__":
    sock = socket.socket()
    sock.bind(('localhost', 44444))
    sock.listen(10)
    users = []
    connect()
```

Клиентская часть (одинаковый код для каждого клиента)

```
from threading import Thread
import socket


def send():
    data = input()
    while data != '/exit':
        sock.send(bytes(data, "utf-8"))
        data = input()
    sock.close()


def receive():
    while True:
        try:
            data = sock.recv(1024)
            print(data.decode('utf-8'))
        except OSError:
            exit()


if __name__ == "__main__":
    sock = socket.socket()
    sock.connect(('localhost', 44444))
    th_send, th_receive = Thread(target=send), Thread(target=receive)
    th_send.start(), th_receive.start()
```

Результат выполнения программы для трех пользователей