Programming Assignment

Part 1: Client Program Report

1. Environment Setup

Prerequisites:

1. **Clang++ (C++ Compiler)**: The program uses the Clang compiler to compile C++ code. Ensure it's installed.
2. **C++14 Support**: The program is written in C++14, so the compiler needs to support C++14 (Clang++ does by default).

2. Compiling the Program

The program is written in C++ and consists of a single file (client2.cpp). To compile it with Clang++, run the following command in your terminal:

```
clang++ -std=c++14 -o client client.cpp
```

or

```
make
```

3. Executing the Program

Once compiled, you can run the executable by executing the following command:

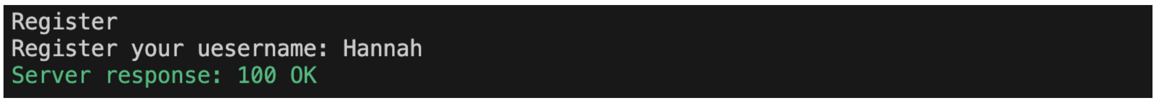
```
./client
```

4. Features and Commands in the Program

The program supports several commands for interaction with a server and other features. Below are the details for the commands, their functions, and how to use them.

A. Register Command

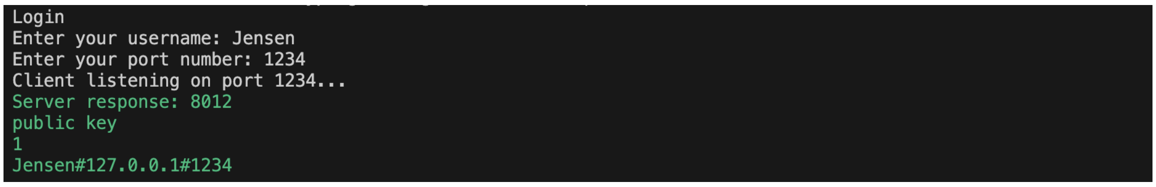
- **Description**: Register your username to the server.
- **Command**: Type Register in the terminal.
- **Functionality**: Prompts the user to enter their **username**.
- **Example**:



```
Register
Register your uesername: Hannah
Server response: 100 OK
```

B. Login Command

- **Description**: Logs the user into the server.
- **Command**: Type Login in the terminal.
- **Functionality**: Prompts the user to enter their **username** and **port number**.Establishes a new listening thread on the provided port and sends the login details to the server.
- **Example**:



```
Login
Enter your username: Jensen
Enter your port number: 1234
Client listening on port 1234...
Server response: 8012
public key
1
Jensen#127.0.0.1#1234
```

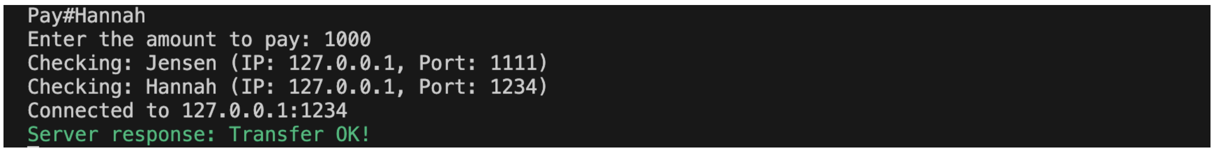
C. Payment Command

- **Description**: Sends a payment request to the server.
- **Command**: Type Pay#<account_name> in the terminal.
- **Functionality**:After typing Pay#<account_name>, you will be prompted to enter the **amount** you want to pay.
- The account name (after Pay#) is used to identify the target account, and the amount is sent as part of the request.
- The server will respond with either a success message or an error code.
- **Example**:

```
Pay#username
```

After this, you will be prompted:

```
Enter the amount to pay:
```



```
Pay#Hannah
Enter the amount to pay: 1000
Checking: Jensen (IP: 127.0.0.1, Port: 1111)
Checking: Hannah (IP: 127.0.0.1, Port: 1234)
Connected to 127.0.0.1:1234
Server response: Transfer OK!
```

C. Exit Command

- **Description**: Exits the program and closes the connection.
- **Command**: Type Exit in the terminal.
- **Functionality**: This will close the program's connection to the server and exit the application.
- **Example**:

```
Exit
```

D. Error Handling

The **client** program includes error-handling mechanisms to manage issues like input format errors, payment failures, and authentication errors.

- **Input Validation**
  - **Login Command**: The program prompts for a **username** and **port number** after the Login command. It checks that both are valid and non-empty.
  - **Payment Command (**Pay#<account_name>**)**:
  - The user must provide a valid **account name** after Pay# and enter a valid **payment amount**.
- **Server and Connection Errors**
  - **Connection Issues**: If the client cannot connect to the server (due to network issues or incorrect port), the program notifies the user.
  - **Port used:** if the port client use to login is already used, system will ask client to login again.
  - **Timeouts**: The user is informed if the server doesn't respond in time.

- **Corrective Actions**
    - The program provides clear feedback for errors, allowing users to retry with corrected details, such as re-entering payment info or login credentials.

E. UI design

- server message: green
- server error message: red
- user input function: white