COP5615 - Distributed Operating System Principles Programming Assignment 1

Bhavana Aleti
bhavanaaleti@ufl.edu
UF ID: 8243-3275

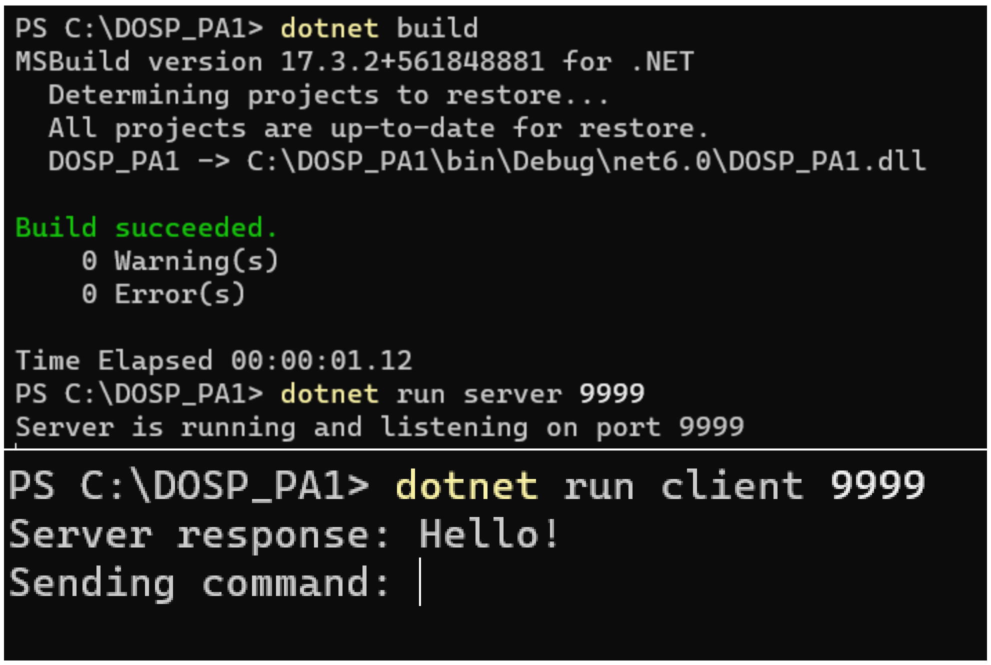
Abhinav Mandala
abhinav.mandala@ufl.edu
UF ID: 2415-2509

Jyothi Santoshi Karuturi
karuturi.j@ufl.edu
UF ID: 2941-1031

Pranay Kumar Reddy Pinninti
p.pinninti@ufl.edu
UF ID: 6539-0190

1 Compile and Run Instructions

- **Software Requirements -** In order to compile and run the application, you must ensure that your system is equipped with .NET version 7.
- Build the code using the command
  dotnet build
- Run the server using the command
  dotnet run server $\langle PortNumber \rangle$
- Run the client using the command
  dotnet run client $\langle PortNumber \rangle$



```
PS C:\DOSP_PA1> dotnet build
MSBuild version 17.3.2+561848881 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  DOSP_PA1 -> C:\DOSP_PA1\bin\Debug\net6.0\DOSP_PA1.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.12
PS C:\DOSP_PA1> dotnet run server 9999
Server is running and listening on port 9999
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command:
```

2 Program Files

- **Team_30.fsproj** - This is a .NET project file, usually named YourProjectName.fsproj, where "YourProjectName" matches your project's name. It stores metadata and configuration settings for your .NET project.
- **Program.fs** - Entry point which initiates server or client with port numbers as input.
- **Server.fsx** - Handles requests from multiple clients and perform arithmetic operations.
- **Client.fsx** - Connects to the server and asynchronously sends and receives the data and handle errors.

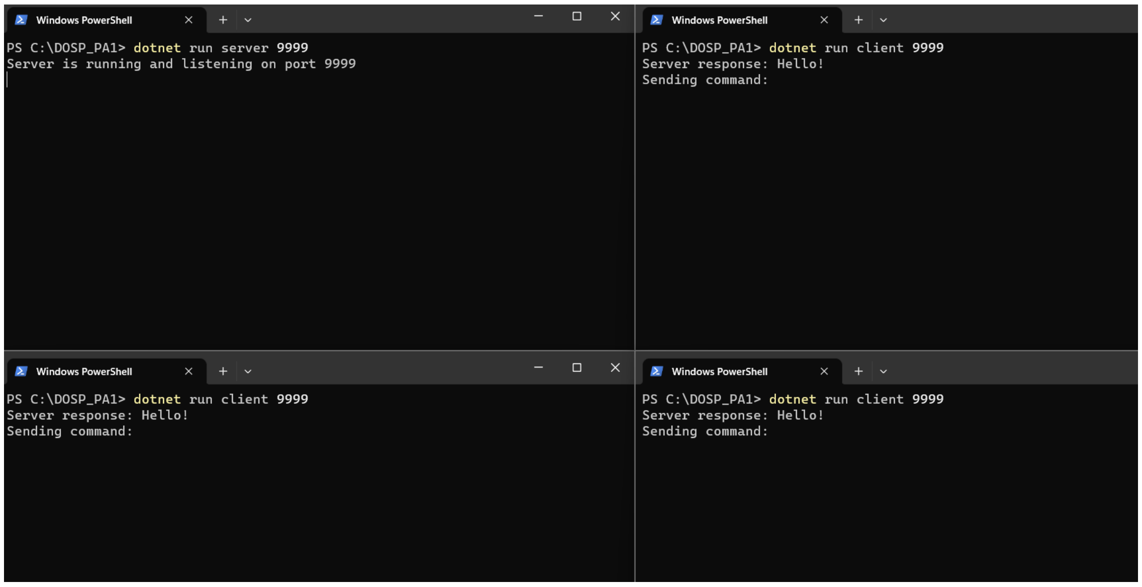
3 Code Structure

- **Server-side Functions:**
  - `mapClientPortToClientId` : Maps client's port number to client ID and stores in a dictionary. This is used to keep track of the client that the server is currently communicating with.
  - `performArithmeticOperations` : Parses and executes mathematical operations from clients, returning results or error codes.
  - `handleClientRequests` : Manages communication with connected clients, processes requests, and sends appropriate responses.
  - `startServer` : Initiates and starts the server, accepts incoming client connections, and manages client tasks until a termination flag is set.
- **Client-side Functions:**
  - `runClient` : Establishes a connection to the server, sets up asynchronous tasks for reading server responses and sending client commands, and handles the termination of the client application.
  - `readFromServerAsync` : Asynchronously reads server responses, translates error codes into descriptive messages, and handles client termination requests.
  - `writeToServerAsync` : Asynchronously captures and sends client input commands to the server while the client application is active.

4 Execution Results

Below are different scenarios of the execution, both positive and negative scenarios. We have provided screenshots for all the scenarios.

Server connected to multiple Clients successfully



```
PS C:\DOSP_PA1> dotnet run server 9999
Server is running and listening on port 9999
```

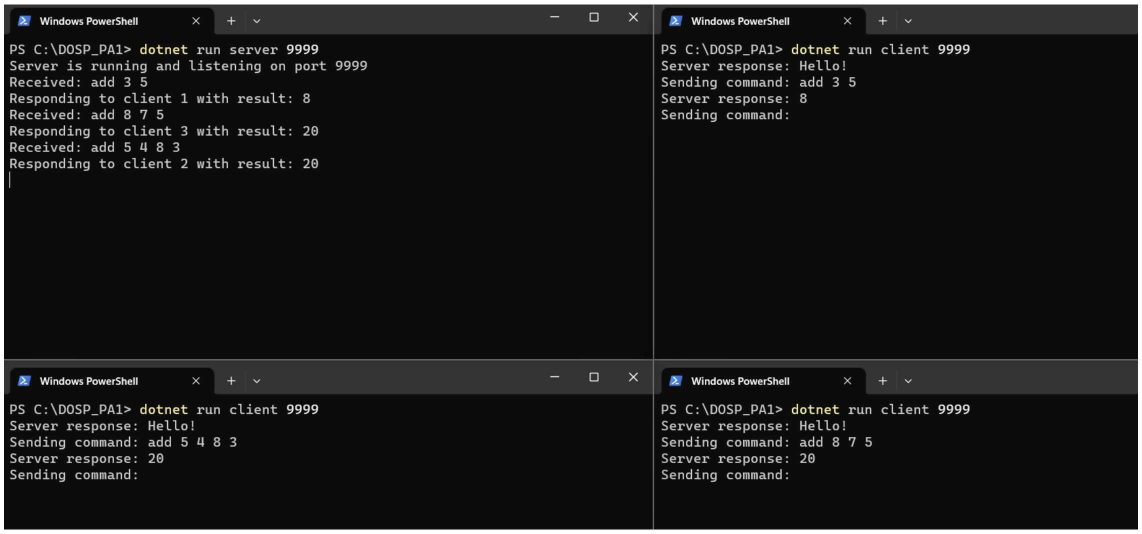
```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command:
```

4.1 Positive Scenarios

- **Add:** Server receives add command for different input lengths from multiple clients.



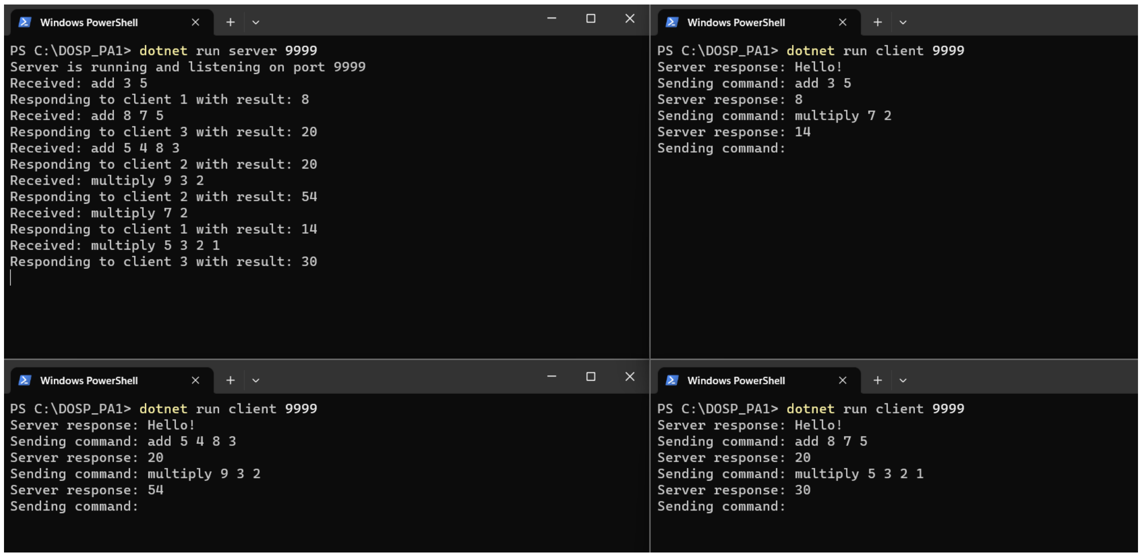
```
PS C:\DOSP_PA1> dotnet run server 9999
Server is running and listening on port 9999
Received: add 3 5
Responding to client 1 with result: 8
Received: add 8 7 5
Responding to client 3 with result: 20
Received: add 5 4 8 3
Responding to client 2 with result: 20
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 5 4 8 3
Server response: 20
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command:
```

- **Multiply:** Server receives multiply command for different input lengths from multiple clients.



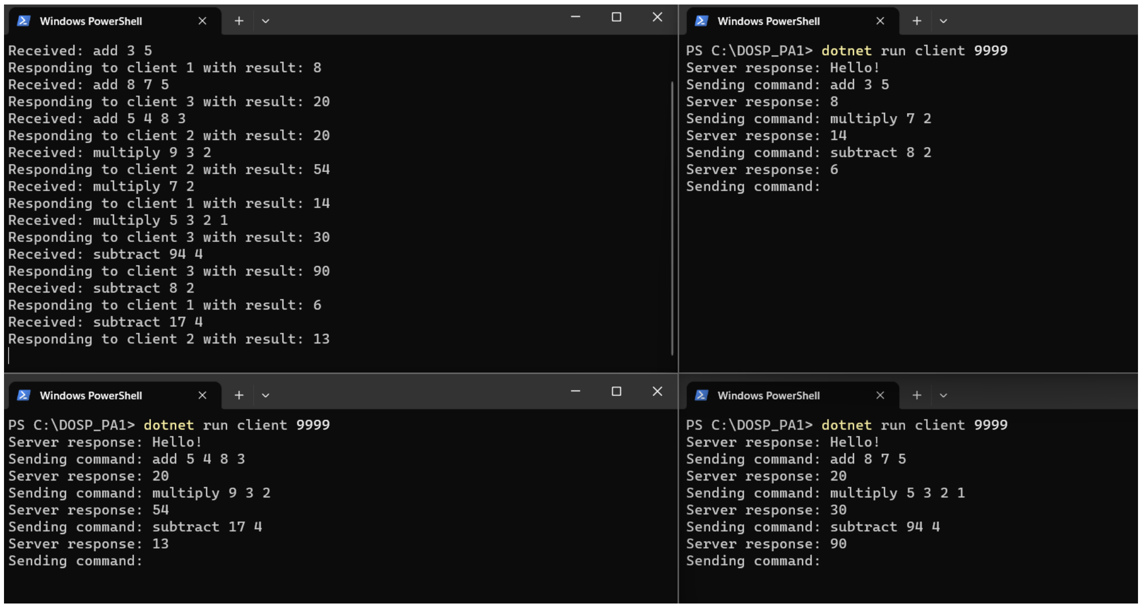
```
PS C:\DOSP_PA1> dotnet run server 9999
Server is running and listening on port 9999
Received: add 3 5
Responding to client 1 with result: 8
Received: add 8 7 5
Responding to client 3 with result: 20
Received: add 5 4 8 3
Responding to client 2 with result: 20
Received: multiply 9 3 2
Responding to client 2 with result: 54
Received: multiply 7 2
Responding to client 1 with result: 14
Received: multiply 5 3 2 1
Responding to client 3 with result: 30
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command: multiply 7 2
Server response: 14
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 5 4 8 3
Server response: 20
Sending command: multiply 9 3 2
Server response: 54
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command: multiply 5 3 2 1
Server response: 30
Sending command:
```

- **Subtract:** Server receives subtract command for different input lengths from multiple clients.



```
Received: add 3 5
Responding to client 1 with result: 8
Received: add 8 7 5
Responding to client 3 with result: 20
Received: add 5 4 8 3
Responding to client 2 with result: 20
Received: multiply 9 3 2
Responding to client 2 with result: 54
Received: multiply 7 2
Responding to client 1 with result: 14
Received: multiply 5 3 2 1
Responding to client 3 with result: 30
Received: subtract 94 4
Responding to client 3 with result: 90
Received: subtract 8 2
Responding to client 1 with result: 6
Received: subtract 17 4
Responding to client 2 with result: 13
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command: multiply 7 2
Server response: 14
Sending command: subtract 8 2
Server response: 6
Sending command:
```

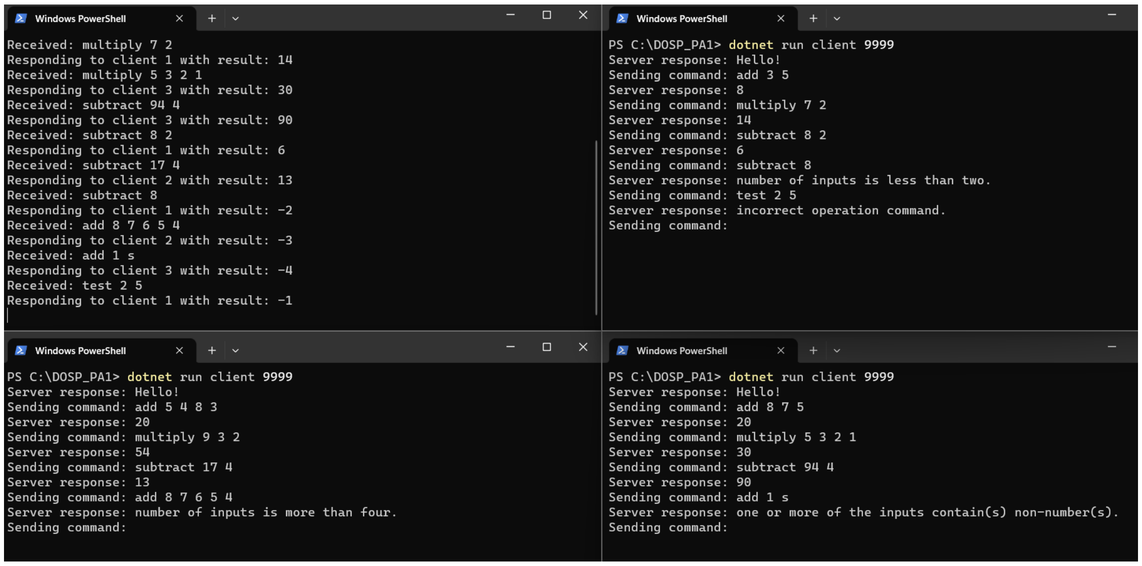
```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 5 4 8 3
Server response: 20
Sending command: multiply 9 3 2
Server response: 54
Sending command: subtract 17 4
Server response: 13
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command: multiply 5 3 2 1
Server response: 30
Sending command: subtract 94 4
Server response: 90
Sending command:
```

4.2 Exception Handling Scenarios

We have handled all different scenarios described in the requirements:

- **Case1:** Server receives number of inputs less than two from client 1, Server returned error code -2 which will be handled in the client code and prints the appropriate error message.
- **Case2:** Server receives number of inputs greater than four from client 2, Server returned the error code -3 which is handled in client and prints the error message according to error code.
- **Case3:** Server receives non-number input (add 1 s) from client 3, error code -4 is returned by server to client, and client prints the error message to the console.
- **Case4:** When an invalid operation command (test 2 5) is provided to server by client 1, error code -1 is returned by server to client, client printed the appropriate message to the console based on the code.



```
Received: multiply 7 2
Responding to client 1 with result: 14
Received: multiply 5 3 2 1
Responding to client 3 with result: 30
Received: subtract 94 4
Responding to client 3 with result: 90
Received: subtract 8 2
Responding to client 1 with result: 6
Received: subtract 17 4
Responding to client 2 with result: 13
Received: subtract 8
Responding to client 1 with result: -2
Received: add 8 7 6 5 4
Responding to client 2 with result: -3
Received: add 1 s
Responding to client 3 with result: -4
Received: test 2 5
Responding to client 1 with result: -1
```

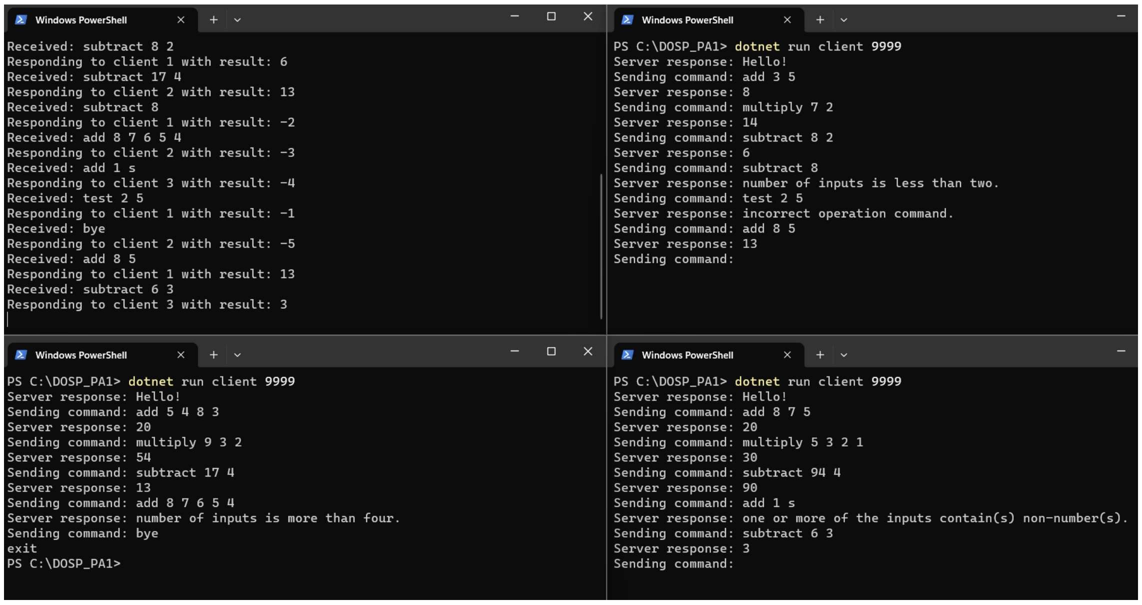
```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command: multiply 7 2
Server response: 14
Sending command: subtract 8 2
Server response: 6
Sending command: subtract 8
Server response: number of inputs is less than two.
Sending command: test 2 5
Server response: incorrect operation command.
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 5 4 8 3
Server response: 20
Sending command: multiply 9 3 2
Server response: 54
Sending command: subtract 17 4
Server response: 13
Sending command: add 8 7 6 5 4
Server response: number of inputs is more than four.
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command: multiply 5 3 2 1
Server response: 30
Sending command: subtract 94 4
Server response: 90
Sending command: add 1 s
Server response: one or more of the inputs contain(s) non-number(s).
Sending command:
```

4.3 Bye Command

Server received command bye from Client 2, server will return error code -5 to client 2, and client 2 will exit after receiving the code from the server. The other clients can still send commands to the server and receive responses.



```
Received: subtract 8 2
Responding to client 1 with result: 6
Received: subtract 17 4
Responding to client 2 with result: 13
Received: subtract 8
Responding to client 1 with result: -2
Received: add 8 7 6 5 4
Responding to client 2 with result: -3
Received: add 1 s
Responding to client 3 with result: -4
Received: test 2 5
Responding to client 1 with result: -1
Received: bye
Responding to client 2 with result: -5
Received: add 8 5
Responding to client 1 with result: 13
Received: subtract 6 3
Responding to client 3 with result: 3
```

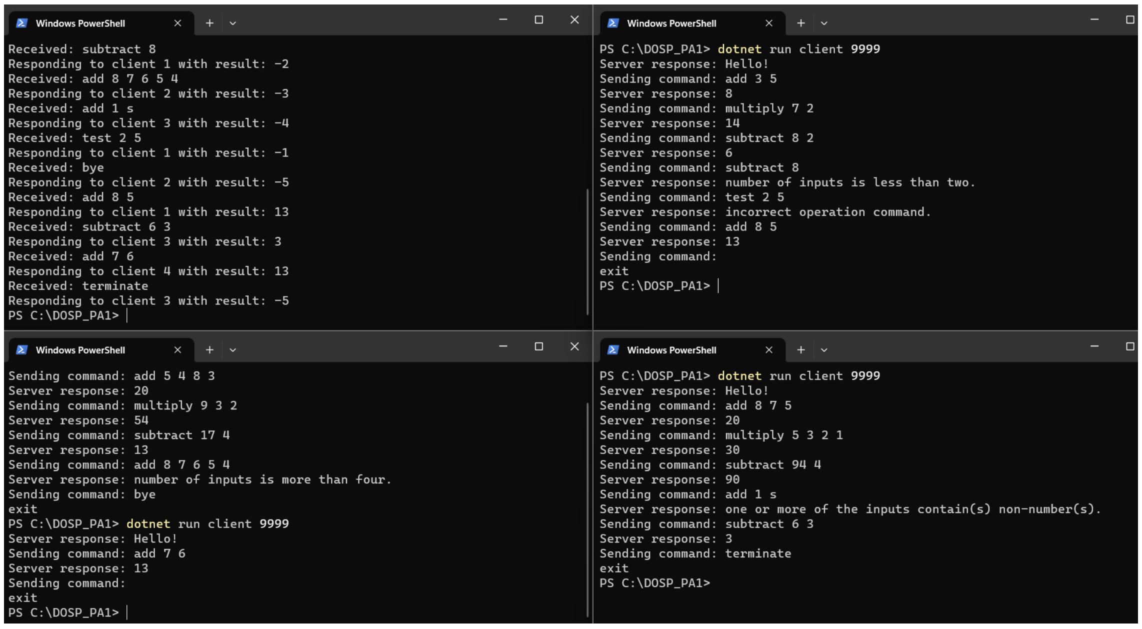
```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command: multiply 7 2
Server response: 14
Sending command: subtract 8 2
Server response: 6
Sending command: subtract 8
Server response: number of inputs is less than two.
Sending command: test 2 5
Server response: incorrect operation command.
Sending command: add 8 5
Server response: 13
Sending command:
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 5 4 8 3
Server response: 20
Sending command: multiply 9 3 2
Server response: 54
Sending command: subtract 17 4
Server response: 13
Sending command: add 8 7 6 5 4
Server response: number of inputs is more than four.
Sending command: bye
exit
PS C:\DOSP_PA1>
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command: multiply 5 3 2 1
Server response: 30
Sending command: subtract 94 4
Server response: 90
Sending command: add 1 s
Server response: one or more of the inputs contain(s) non-number(s).
Sending command: subtract 6 3
Server response: 3
Sending command:
```

4.4 Terminate Command

Server received command terminate from Client 3, server will return error code -5 to client 3, and client 3 will exit after receiving the code from the server. And all the other available clients which are 1 and 4 will also exit, as it receives command from server. Once all the clients are terminated, the server is also terminated.



```
Received: subtract 8
Responding to client 1 with result: -2
Received: add 8 7 6 5 4
Responding to client 2 with result: -3
Received: add 1 s
Responding to client 3 with result: -4
Received: test 2 5
Responding to client 1 with result: -1
Received: bye
Responding to client 2 with result: -5
Received: add 8 5
Responding to client 1 with result: 13
Received: subtract 6 3
Responding to client 3 with result: 3
Received: add 7 6
Responding to client 4 with result: 13
Received: terminate
Responding to client 3 with result: -5
PS C:\DOSP_PA1>
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 3 5
Server response: 8
Sending command: multiply 7 2
Server response: 14
Sending command: subtract 8 2
Server response: 6
Sending command: subtract 8
Server response: number of inputs is less than two.
Sending command: test 2 5
Server response: incorrect operation command.
Sending command: add 8 5
Server response: 13
Sending command:
exit
PS C:\DOSP_PA1>
```

```
Sending command: add 5 4 8 3
Server response: 20
Sending command: multiply 9 3 2
Server response: 54
Sending command: subtract 17 4
Server response: 13
Sending command: add 8 7 6 5 4
Server response: number of inputs is more than four.
Sending command: bye
exit
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 7 6
Server response: 13
Sending command:
exit
PS C:\DOSP_PA1>
```

```
PS C:\DOSP_PA1> dotnet run client 9999
Server response: Hello!
Sending command: add 8 7 5
Server response: 20
Sending command: multiply 5 3 2 1
Server response: 30
Sending command: subtract 94 4
Server response: 90
Sending command: add 1 s
Server response: one or more of the inputs contain(s) non-number(s).
Sending command: subtract 6 3
Server response: 3
Sending command: terminate
exit
PS C:\DOSP_PA1>
```

5 Additional Comments

5.1 Code Organization:

The code has been thoughtfully structured with the use of multiple small functions to enhance readability and maintainability. This approach makes the codebase more organized and comprehensible. Each function is designed to perform a specific task or encapsulate a particular piece of logic, making it easier to understand and debug. Furthermore, this modular design allows for efficient code reuse, as these small functions can be utilized in various parts of the application. In summary, the use of small functions significantly improves the overall quality of the code by promoting clarity, maintainability, and reusability.

5.2 Error Handling:

Our code includes some error handling for known exceptions to handle unexpected errors gracefully. Some of them include

- **Server Not Running:** Our code gracefully handles scenarios where clients attempt to connect to a non-existent or unresponsive server, preventing wasted connection attempts and informing users of the server's unavailability.
- **Invalid Operands and Operators:** The code includes validation to check for invalid operands and operators provided by clients, ensuring that calculations proceed smoothly. It captures number format exceptions and responds with clear error messages, guiding users to correct their input.

5.3 Extensibility:

The following enhancements can be implemented in our code -

- **File Transfer:** Enable clients to exchange files by introducing commands such as "sendfile" and "receivefile." The server manages file storage and facilitates secure data transfer among clients.
- **Peer-to-Peer Communication:** Implement direct client-to-client interaction, enabling users to connect without server intervention. Clients can discover and establish connections with each other using peer-to-peer protocols.
- **Chat Rooms and Group Messaging:** Establish virtual chat spaces where multiple clients can join and communicate in real-time. Implement group messaging functionalities, including user administration and message history tracking.

6 Contributions

This project was a collaborative effort, with each team member contributing to the development of both the server and client components. We have brainstormed the design and the responsibilities were distributed equally among the four of us.

- **Server Program Development:**
  - `Abhinav Mandala` : Developed the server program, focusing on establishing and managing TCP/IP socket connections and also implemented the logic for handling concurrent client requests by spawning asynchronous tasks for each connection.
  - `Jyothi Santoshi Karuturi` : Contributed to the server program's core functionality, particularly in processing and responding to client commands made sure server could handle various operation commands and generate appropriate responses.
- **Client Program Development:**
  - `Bhavana Aleti` : Developed the client program, including the asynchronous communication with the server. Implemented the logic for sending messages to the server, receiving responses, and printing them out.
  - `Pranay Kumar Reddy Pinninti` : Collaborated on the client program's development, focusing on user interaction and input handling. Implemented the logic for processing user commands, sending them to the server, and displaying the server's responses.
- **Exception Handling:**
  - `All Team Members` : We all collaboratively designed and implemented the error handling for the server to handle unexpected inputs. Ensured that error codes were appropriately generated for incorrect commands or invalid inputs, following the specified code list in the requirement.
- **Testing:**
  - `All Team Members` : All of us participated in extensive testing, ensuring the reliability and correctness of the server-client interaction.
- **Report:**
  - `All Team Members` : We divided the report sections among us and ensured the document has all the information required.