**III.1.3. Trả lời câu hỏi:**

1. Khi telnet vào Server bằng port 5000 tại sao port client dùng để kết nối vào lại không phải là port 5000?

Port client sẽ được tạo tự động như một định danh cho mỗi client để server có thể giao tiếp chính xác với client.

Port server hoàn là dùng để cho các client kết nối.

2) Như hình trên, giải thích vì sao có client dùng port giá trị 1699 ?

Vì đây là port dùng để gán cho các client để server có thể giao tiếp cho client thông qua port của client.

3) Có phải lúc nào client cũng mở port 1699 để kết nối với Server không ?

Không phải lúc nào client cũng mỡ port 1699 để kết nối với Server. Port của client sẽ tạo ngẫu nhiên từ 0 🡪 65535

4) Ta có thể chạy 2 chương trình CMD để telnet vào Server trên được không ? Giải thích.

Ta có thể chạy 2 chương trình CMD để telnet vào Server trên được. Khi chạy 2 chương trình CMD, do chạy trên cùng một máy nên 2 chương trình sẽ chạy chung một port khi giao tiếp với server

**III.2.3. Trả lời câu hỏi:**

1. Tại sao trong đoạn code viết thêm vào ta không cần phải khởi tạo độ lớn ban đầu cho buffer ?

Trong đoạn code viết thêm vào ta không cần phải khởi tạo độ lớn ban đầu cho buffer vì chỉ cần gửi dữ liệu cho client chứ không nhận dữ liệu từ client nên không cần phải khởi tạo độ lớn ban đầu. Mặc khác, tham số truyền vào đã có độ dài của mảng.

1. Giải thích ý nghĩa các tham số của phương thức clientSocket.Send.

Cú pháp: .Send(Buff, Offset, Size, Socket Flags)

**Buff:** dùng để gửi dữ liệu

**Offset:** Mảng các byte để chứa dữ liệu

**Size:** Vị trí trong mảng cần gửi

**SocketFlags:** Cách gửi dữ liệu trên Socket

**III.3.3. Trả lời câu hỏi:**

1. Khi client kết nối đến server mà server chưa chạy hoạc là đường truyền mạng bị rớt thì có hiện tượng gì xảy ra ?, giải thích.

Xảy ra lỗi: Không thể kết nối với máy chủ và không tìm thấy máy chủ để kết nối. Vì khi client kết nối tới máy chủ thì máy chủ phải được khởi động trước và phải có kết nối ổn định để client có thể giao tiếp được với server.

1. Từ đoạn code ở chương trình client:

Nếu ta thay byteReceive bằng buff.Length thì kết quả câu nhận được sẽ hiển thị theo đúng số kích thước mảng buff đã được chỉ định. Vì khi gửi dữ liệu giữa server và client, dữ liệu sẽ được lưu ở bộ nhớ đệm với đúng số byte mà server đã gửi. Khi sử dụng buff.Length thì client sẽ lấy trong bộ nhớ đệm số byte theo đúng kích thước của buff.Length.

**6.3. Trả lời câu hỏi:**

1) Khi thay đổi kích thước bộ đệm từ 1024 xuống 10 thì có hiện tượng gì xảy ra ?. Giải thích

=> Khi thay đổi kích thước bộ đệm từ 1024 xuống 10 thì dữ liệu truyền từ server gửi tới client hay từ client tới server sẽ chỉ nhận được giới hạn là 10 byte dữ liệu nên sẽ có sự sai lệch về dữ liệu khi truyền giữa client với server.

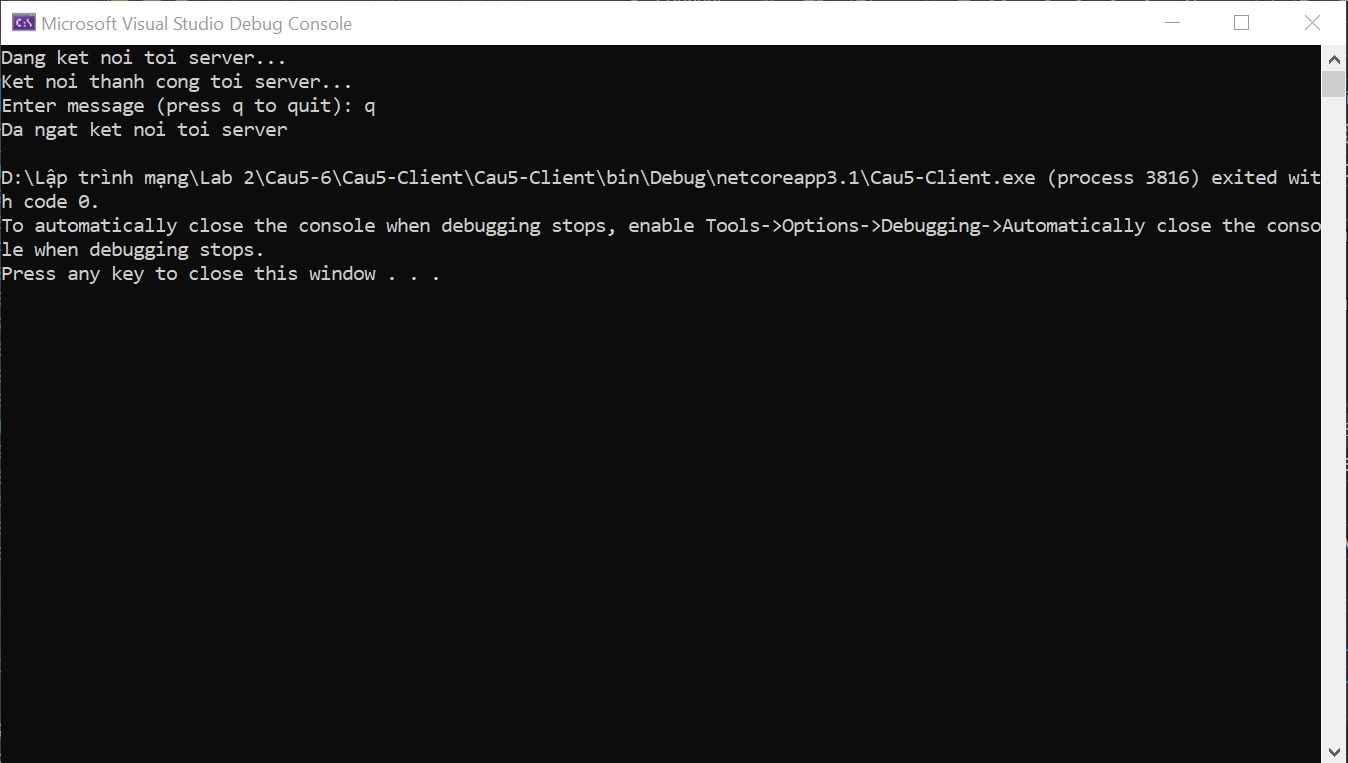
Nếu giới hạn byte chỉ có 10 thì dữ liệu nhận được chỉ có 10 byte. Các dữ liệu còn lại vẫn được lưu trong bộ nhớ đệm nếu client gửi dữ liệu có số byte lớn hơn 10.

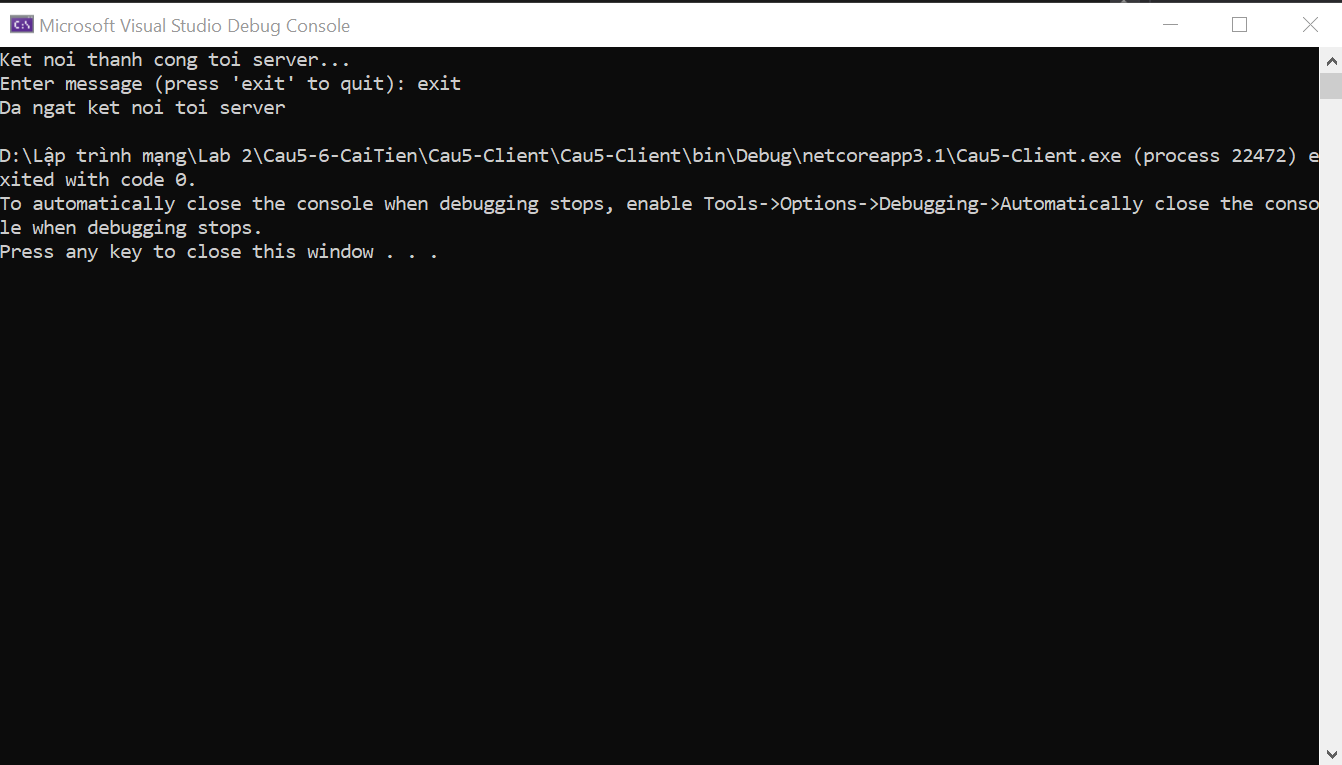
2) Nếu ta không reset lại bộ đệm thì có hiện tượng gì xảy ra? Giải thích

Nếu không reset lại bộ đệm thù trong bộ đệm vẫn còn lưu dữ liệu bên trong nếu như client gửi dữ liệu có số byte quá lớn. Điều này dẫn đến dữ liệu từ server gửi về client bị sai lệch.

**IV. Bài tập**

1. Chương trình trên khi tắt client thì trên server sẽ bị lỗi, hãy cải tiến lại chương để khi client tắt đột ngột trên server sẽ thông báo client đã ngắt kết nối

2. Cải tiến chương trình: khi client nhập chữ “exit” thì sẽ tắt client, trên server thông báo  
client đã ngắt kết nối



1. Viết chương trình thực hiện các phép tính số học, việc thực hiện tính toán được thực  
   hiện trên server, phía client sẽ nhập vào các toán hạng và các phép tính

