COVER

- Programming Assignment 1
- CIS427
- Ray Sahi, Max Afanasyev
- Winter 2022

Table of Contents:

Section 1:

Introduction:

- Ray has started the project early on, which gave us some advantages. I (Max) joined in over the weekend and we started working on it together. Ray was able to perfectly implement the client and the server, so they can communicate with each other. I helped out with the code for each of the functions. We, then, both worked on the README file.

Instructions:

- To compile our client and server, you need to first grab the client.cpp and server.cpp files, which are found within the umd.login directory.
- Once you have done that, use gcc -o “filename” for Server.cpp and Client.cpp to compile the files. You can also use a Visual Studio compiler to run this.
- The files that have open access are Server.cpp and Client.cpp, the solution file will not be runnable and will only run when the .cpp file is looking to run.

Issues:

- We did run into some issues with our server and client. Our list function would not output to the client (please see our coding comments) however, would output completely correctly to the server. We were stuck trying to send the output.txt file as one buffer back to the client, but this not working despite our best efforts.
- Overall, everything works perfectly, no other issues.

Section 2:

Test Cases:

[Test Cases for your code – any input from the “smart user”. These are not the questions.]

Test #	Valid / Invalid Data	Description of test	Input Value	Expected Output	Actual Output	Test Pass / Fail
1	Valid	Testing add function	add Ray Sahi 123-123-1234	1001, Ray Sahi, 123-123-1234	1001, Ray Sahi, 123-123-1234	Pass
2	Valid	Testing Delete function	delete 1001	No 1001 record	No 1001 record	Pass
3	Valid	Testing List Function	list	1002 Max A 1003 John Doe	1002 Max A 1003 John Doe	Pass
4	Valid	Testing Shutdown Function	shutdown	Shutdown server	Shutdown server	Pass
5	Valid	Testing Quit Function	quit	quit	qu	pass

Test Case #1:

```
Powerhouse successfully connected on port 20021
The new Record ID is 1001
The new Record ID is 1002
The new Record ID is 1003
```

```
List of Commands:
#1: Type 'add' followed by 'first name' 'last name' and 'phone number' to add an employee.
#2: Type 'delete' followed by an 'id' to delete an employee.
#3: Type 'list' to list all employees.
#4: Type 'shutdown' to shut down server.
#5: Type 'quit' to disconnect from server.
> add Ray Sahi 123-123-1234
SERVER> 200 OK
> add Max A 321-321-4321
SERVER> 200 OK
> add John Doe 412-421-4213
SERVER> 200 OK
>
```

Test Case #2:

```
Powerhouse successfully connected on port 20043
200 OK
```

```
List of Commands:
#1: Type 'add' followed by 'first name' 'last name' and 'phone number' to add an employee.
#2: Type 'delete' followed by an 'id' to delete an employee.
#3: Type 'list' to list all employees.
#4: Type 'shutdown' to shut down server.
#5: Type 'quit' to disconnect from server.
> delete 1001
SERVER> 200 OK
>
```

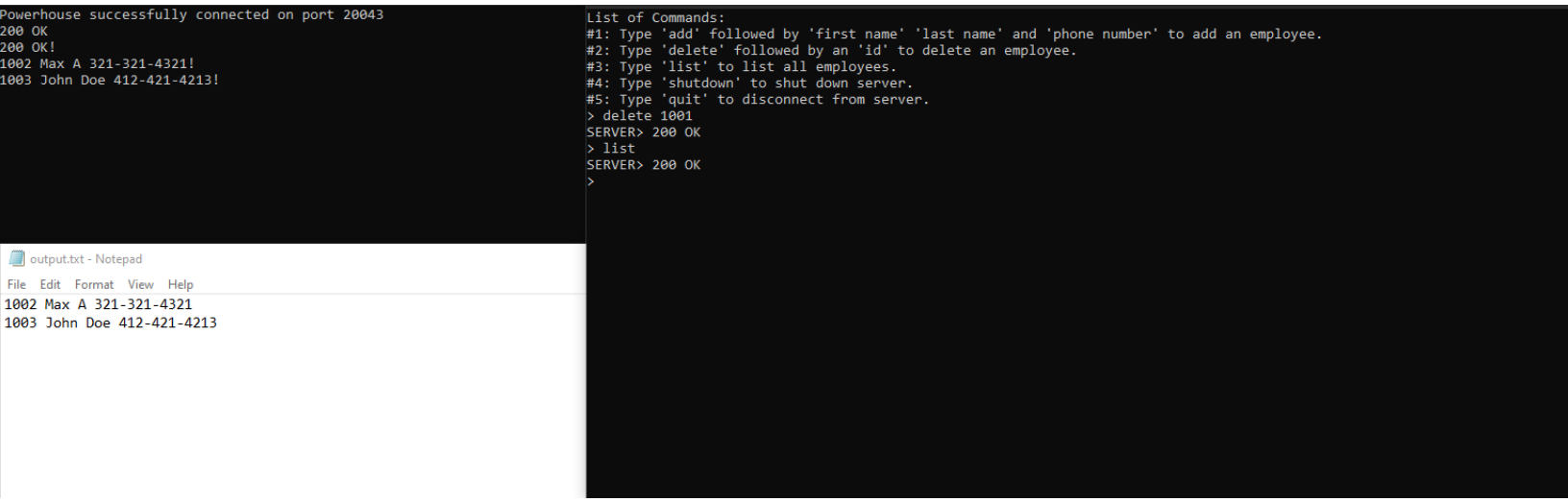
output.txt - Notepad

File Edit Format View Help

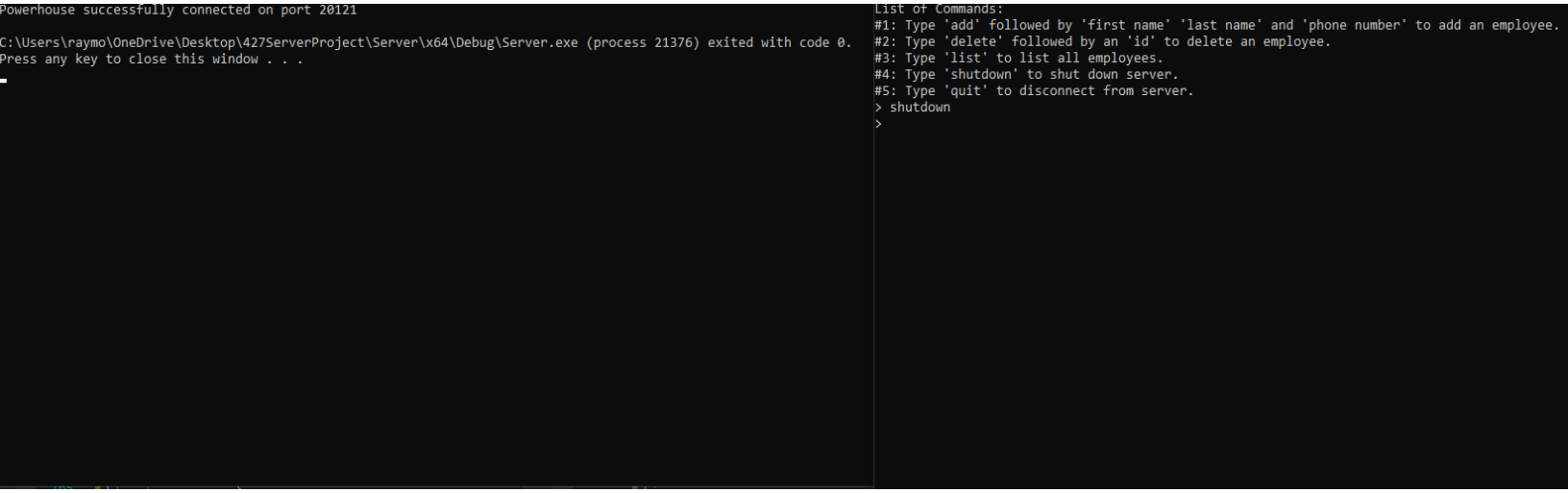
```
1002 Max A 321-321-4321
1003 John Doe 412-421-4213
```

Ln 3, Col 1 | 100% | Windows (CRLF) | UTF-8

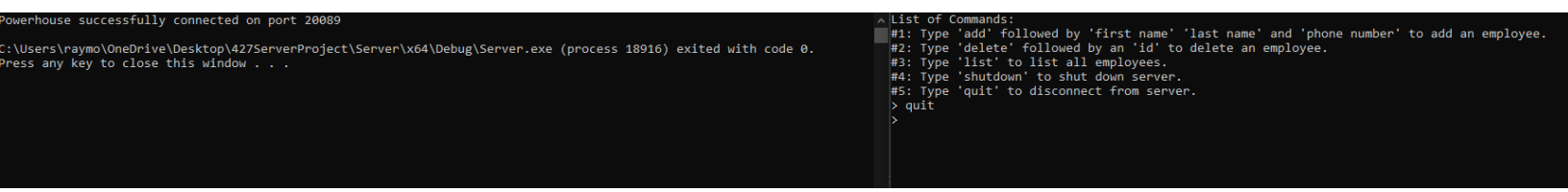
Test Case #3:



Test Case #4:



Test Case #5:



Section 3:

Source Code (Server.cpp):

```
#include <iostream>
#include <Winsock2.h>
#include <fstream>
#include <sstream>
#include <string>
#include <cstring>
#include <WS2tcpip.h>
#pragma comment (lib, "ws2_32.lib")
#define SERVER_PORT 8645
#define BUFFER 4096

using namespace std;

void main()
{
	// Initilize winsock
   WSADATA WsData;
	WORD ver = MAKEWORD(2, 2);
	int wsOk = WSAStartup(ver, &WsData);

	if (wsOk != 0) {
		cerr << "Cant initialize winsock!" << endl;
		return;
	}
```

```
// Create a socket
struct sockaddr_in sin;
SOCKET listening = socket(AF_INET, SOCK_STREAM, 0);
if (listening == INVALID_SOCKET) {
	cerr << "Cant create a socket!" << endl;
	return;
}

// Bind socket to ip and port
sockaddr_in hint;
hint.sin_family = AF_INET;
hint.sin_port = htons(SERVER_PORT);
hint.sin_addr.S_un.S_addr = INADDR_ANY;
bind(listening, (sockaddr*)&hint, sizeof(hint));

// Tell winsock the socket is for listening
listen(listening, SOMAXCONN);

// Wait for a connection
sockaddr_in client;
int clientSize = sizeof(client);

SOCKET clientSocket = accept(listening, (sockaddr*)&client, &clientSize);
if (clientSocket == INVALID_SOCKET) {
	cerr << "Cannot bind client socket!" << endl;
	return;
}
```

```
	char host[NI_MAXHOST]; // Clients remote name
	char service[NI_MAXHOST]; // Service (i.e. port) the client is connect on

	ZeroMemory(host, NI_MAXHOST); // For Mac
	ZeroMemory(service, NI_MAXHOST); // For Mac

	if (getnameinfo((sockaddr*)&client, sizeof(client), host, NI_MAXHOST, service, NI_MAXSERV, 0) == 0) {
		cout << host << " successfully connected on port " << service << endl;
	}
	else {
		inet_ntop(AF_INET, &client.sin_addr, host, NI_MAXHOST);
		cout << host << "connected on port" << ntohs(client.sin_port) << endl;
	}

	// Close listening socket
	closesocket(listening);

	// Variable Declarations

	char buf[BUFFER]; //Buffer / Message
	fstream file; // Output File
	ofstream temp; // Temp output file
	ifstream save; // Inputing output file
	int idnum = 1000; // Id#
	bool loop_control = true; // Loop Controller
```

```
// While loop: accept and echo message back to client / Functions
while (loop_control) {
	ZeroMemory(buf, BUFFER);

	// Wait for client to send data
	int bytesReceived = recv(clientSocket, buf, BUFFER, 0);
	if (bytesReceived == SOCKET_ERROR) {
		cerr << "Error in recv()" << endl;
		break;
	}
	if (bytesReceived == 0) {
		cout << "Client Disconnected" << endl;
		break;
	}

	string first, second, third, fourth, fifth, total, fixed, newSecond, space = " "; //
Each Word
	stringstream in, out; // Streaming String
	string clientIn = string(buf, 0, bytesReceived);
	in.str(clientIn);
	in >> first >> second >> third >> fourth >> fifth;

	// Add Function
		if (first == "add") {
			idnum++;
			file.open("output.txt", std::ios_base::app);
```

```
                    file << idnum << space << second << space << third << space <<
fourth << space << fifth << endl;
                    cout << "The new Record ID is " << idnum << endl;
                    send(clientSocket, "200 OK", 7, 0);
                    file.close();
                }

            // Delete Function
                else if (first == "delete") {
                    int idcompare = stoi(second); // Create integer out of string

                    if (idcompare < 0) {
                        cout << "403 ID Invalid" << endl;
                        send(clientSocket, "403 ID Invalid", 14, 0);
                        }

                    temp.open("temp.txt"); // Open a temp file
                    file.open("output.txt"); // Open the writing file
                    while (getline(file, fixed)) {
                        out.str(fixed);
                        out >> newSecond;
                        if (newSecond != second) {
                            temp << fixed << endl;
                        }
                    }
                    file.close();
                    temp.close();
```

```
                remove("output.txt");
                rename("temp.txt", "output.txt");
                cout << "200 OK " << endl;
                send(clientSocket, "200 OK", 7, 0);
        }

    // List Function
        else if (first == "list") {
                string line;
                file.open("output.txt");
                getline(file, line);

                if (!line.empty()) {
                        string one, two, three, four, total;
                        cout << "200 OK!" << endl;
                        send(clientSocket, "200 OK", 7, 0);
                        file.close();
                        file.open("output.txt");

                        while (file >> one >> two >> three >> four) {
                                total = one + space + two + space + three + space +
four + "!";
                                cout << total << endl;
                                strcpy_s(buf, total.c_str());
                                /* send(clientSocket, buf, 50, 0); This is not sending
the output back to client for some reason,
                                the client needs more recv. This is causing output
errors on the client side. A send() function needs to be used in order
```

```
                                        to get the correct output.
                                        */
                                }
                                file.close();
                        }
                        else {
                                cout << "The list is empty." << endl;
                                send(clientSocket, "The list is empty!", 19, 0);
                        }
                }

        // Shutdown Function
                else if (first == "shutdown") {
                        loop_control = false;
                        return;
                }
        // Quit Function
                else if (first == "quit") {
                        loop_control = false;
                        closesocket(clientSocket); // Close current socket
                        bind(listening, (sockaddr*)&hint, sizeof(hint)); // Tell socket to
bind again
                        listen(listening, SOMAXCONN); // Listen for socket
                        return;
                }
                else {
                        return;
```

```
                    }

        }

        // Close the socket
        closesocket(clientSocket);

        // Shutdown winsock
        WSACleanup();
}
```

Source Code (Client.cpp)

```
#include <iostream>
#include <string>
#include <WS2tcpip.h>
#pragma comment(lib, "WS2_32.lib")
#define SERVER_PORT 8645
#define BUFFER 4096

using namespace std;

void main() {

	string ipAddress = "127.0.0.1"; // Ip address of server
	int port = SERVER_PORT;

	// Initialize winsock
	WSADATA data;
	WORD ver = MAKEWORD(2, 2);
	int wsResult = WSAStartup(ver, &data);
	if (wsResult != 0) {
		cerr << "Cant start winsock! Error #" << wsResult << endl;
		return;
	}

	// Create Socket
	SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
	if (sock == INVALID_SOCKET) {
```

```
		cerr << "Cannot create socket!" << endl;
		WSACleanup();
		return;
	}

	// Fill in hint structure
	sockaddr_in hint;
	hint.sin_family = AF_INET;
	hint.sin_port = htons(SERVER_PORT);
	inet_pton(AF_INET, ipAddress.c_str(), &hint.sin_addr);

	// Connect to server
	int connection = connect(sock, (sockaddr*)&hint, sizeof(hint));
	if (connection == SOCKET_ERROR) {
		cerr << "Cant connect to server!" << endl;
		closesocket(sock);
		WSACleanup();
		return;
	}

	// Do While loop to send and recieve data
	char buf[BUFFER];
	string userInput;
	cout << "List of Commands:" << endl;
	cout << "#1: Type 'add' followed by 'first name' 'last name' and 'phone number' to add an employee. " << endl;
	cout << "#2: Type 'delete' followed by an 'id' to delete an employee." << endl;
```

```
	cout << "#3: Type 'list' to list all employees." << endl;
	cout << "#4: Type 'shutdown' to shut down server. " << endl;
	cout << "#5: Type 'quit' to disconnect from server. " << endl;

	do {
		// Prompt user for text
		cout << "> ";
		getline(cin, userInput);

		if (userInput.size() > 0) {  // Make sure user typed something in
			// Send text
			int sendResult = send(sock, userInput.c_str(), userInput.size() + 1, 0);
			if (sendResult != SOCKET_ERROR) {

				// Wait for response
				ZeroMemory(buf, BUFFER);
				int bytesReceived = recv(sock, buf, BUFFER, 0);

				if (bytesReceived > 0) {
					// Echo response to console
					cout << "SERVER> " << string(buf, 0, bytesReceived) <<
endl;
				}
			}
		}

	} while (userInput.size() > 0);
```

```
        // Close down client
        closesocket(sock);
        WSACleanup();
}
```