FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES AND MULTIMEDIA ARTS

CCS0051

(Parallel and Distributed Computing)

FINAL PROJECT 2

Student Name / Group Name:	Group 1
Members (if Group):	<table><tr><th>Name</th><th>Role</th></tr><tr><td>Centeno, Joaquin Franco T.</td><td></td></tr><tr><td>Reyes, Jan Andrei M.</td><td></td></tr><tr><td>Robledo, Ian Justine R.</td><td></td></tr></table>
Section:	TN22
Professor:	Doc. Hadji Tejuco

NSTALL INSTRUCTIONS

Note: For me, copying and pasting code from outside of the VM does not work. To work around this, open this GitHub repository from within the VM itself then copy it into the .cpp files.

Note: For me, selecting the entirety of the code and using CTRL + K to cut all at once does not work. Instead, you can select the first line of your code and hold CTRL + K to erase the existing contents to paste in the new updated code from GitHub. (If you find a work-around please tell us because I legit have no idea what I'm doing lmao)

Note: You can press CTRL + L to "clear" the terminal. Makes it easier to read. (All it does is move the cursor down a few lines lmao)

1. Open terminal and type in the following:

```
mkdir pusoy_clash

cd pusoy_clash
```

mkdir creates a new directory called pusoy_clash and uses that directory using cd

2. To create the two files: server and client

```
nano server.cpp
```

L After you paste in the code, do CTRL + O then ENTER to save the file and CTRL + X to exit

L Do the same for the next file

```
nano client.cpp
```

To run (compile) files

Open terminal and do the following:

1. For server terminal:

```
cd pusoy_clash
  g++ server.cpp -o server -pthread
```

2. For client terminal

```
cd pusoy_clash
  g++ client.cpp -o client -pthread
```

Note: You need only run the server.cpp once for the 1 server terminal. For the client terminals, the program will not begin unless 4 players are connected, meaning you need to have 4 seperate terminals running with client.cpp to start the program.

Limitations

There is currently no "retry" function, so you have to restart the entire process of recompiling the files again.

Might be some bugs, idk

SOURCE CODES

client.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

#ifdef _WIN32
    #include <winsock2.h>
    #include <ws2tcpip.h>
    #pragma comment(lib, "ws2_32.lib")
#else
    #include <unistd.h>
    #include <arpa/inet.h>
    #include <sys/socket.h>
#endif

const int PORT = 12345;

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;

    sock = socket(AF_INET, SOCK_STREAM, 0);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    cout << "Enter your name: ";
    string name;
    getline(cin, name);

    inet_pton(AF_INET, "192.168.254.143", &serv_addr.sin_addr); // Replace IP
if needed

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        cerr << "Connection failed\n";
        return 1;
    }

    send(sock, name.c_str(), name.size(), 0);

    char buffer[1024];
```

```
    while (true) {
        int valread = read(sock, buffer, sizeof(buffer) - 1);
        if (valread <= 0) break;
        buffer[valread] = '\0';
        cout << buffer;
    }

    close(sock);
    return 0;
}
```

server.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <thread>
#include <cstdlib>
#include <ctime>
#include <algorithm>
using namespace std;

#ifdef _WIN32
    #include <winsock2.h>
    #include <ws2tcpip.h>
    #pragma comment(lib, "ws2_32.lib")
#else
    #include <unistd.h>
    #include <arpa/inet.h>
    #include <netdb.h>
    #include <netinet/in.h>
    #include <sys/socket.h>
#endif

const int PORT = 12345;
const int MAX_CLIENTS = 4;

vector<int> client_sockets;
vector<string> playerNames(MAX_CLIENTS);
vector<int> playerScores(MAX_CLIENTS, 0);

void broadcast(const string &message) {
```

```
    for (int sock : client_sockets) {
        send(sock, message.c_str(), message.size(), 0);
    }
}

void handleGame() {
    srand(time(0));
    string startMsg = "\n=== Pusoy Clash Game Simulation ===\n";
    broadcast(startMsg);
    sleep(1);

    for (int round = 1; round <= 3; ++round) {
        broadcast("\n--- Starting Round " + to_string(round) + " ---\n");
        vector<int> cards(MAX_CLIENTS);

        for (int i = 0; i < MAX_CLIENTS; ++i) {
            cards[i] = rand() % 13 + 2;
            broadcast(playerNames[i] + " draws a card: " + to_string(cards[i])
+ "\n");
            sleep(1);
        }

        int maxCard = *max_element(cards.begin(), cards.end());
        vector<int> winners;
        for (int i = 0; i < MAX_CLIENTS; ++i) {
            if (cards[i] == maxCard) {
                winners.push_back(i);
            }
        }

        if (winners.size() == 1) {
            playerScores[winners[0]]++;
            broadcast("\nRound " + to_string(round) + " winner: " +
playerNames[winners[0]] +
                      " with card " + to_string(maxCard) + "!\n");
        } else {
            broadcast("\nRound " + to_string(round) + " is a tie!\n");
        }
        sleep(2);
    }

    broadcast("\n=== Final Scores ===\n");
    for (int i = 0; i < MAX_CLIENTS; ++i) {
```

```
        broadcast(playerNames[i] + ": " + to_string(playerScores[i]) + "
points\n");
    }

    int maxScore = *max_element(playerScores.begin(), playerScores.end());
    vector<int> finalWinners;
    for (int i = 0; i < MAX_CLIENTS; ++i) {
        if (playerScores[i] == maxScore) {
            finalWinners.push_back(i);
        }
    }

    if (finalWinners.size() == 1) {
        broadcast("\n" + playerNames[finalWinners[0]] + " won with " +
                  to_string(maxScore) + " points!\n");
    } else {
        broadcast("\nIt's a tie between:\n");
        for (int i : finalWinners) {
            broadcast("- " + playerNames[i] + "\n");
        }
    }
}

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    socklen_t addrlen = sizeof(address);

    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt));
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);
    bind(server_fd, (struct sockaddr *)&address, sizeof(address));
    listen(server_fd, MAX_CLIENTS);

    cout << "Server started on port " << PORT << ". Waiting for players...\n";

    while (client_sockets.size() < MAX_CLIENTS) {
        new_socket = accept(server_fd, (struct sockaddr *)&address, &addrlen);
```

```
        char nameBuffer[1024] = {0};
        read(new_socket, nameBuffer, sizeof(nameBuffer));
        string playerName(nameBuffer);

        client_sockets.push_back(new_socket);
        playerNames[client_sockets.size() - 1] = playerName;

        string joinMsg = playerName + " has joined the game!\n";
        cout << joinMsg;
        broadcast(joinMsg);
    }

    broadcast("\nAll players connected. Game is starting!\n");
    handleGame();

    for (int sock : client_sockets) close(sock);
    close(server_fd);
    return 0;
}
```

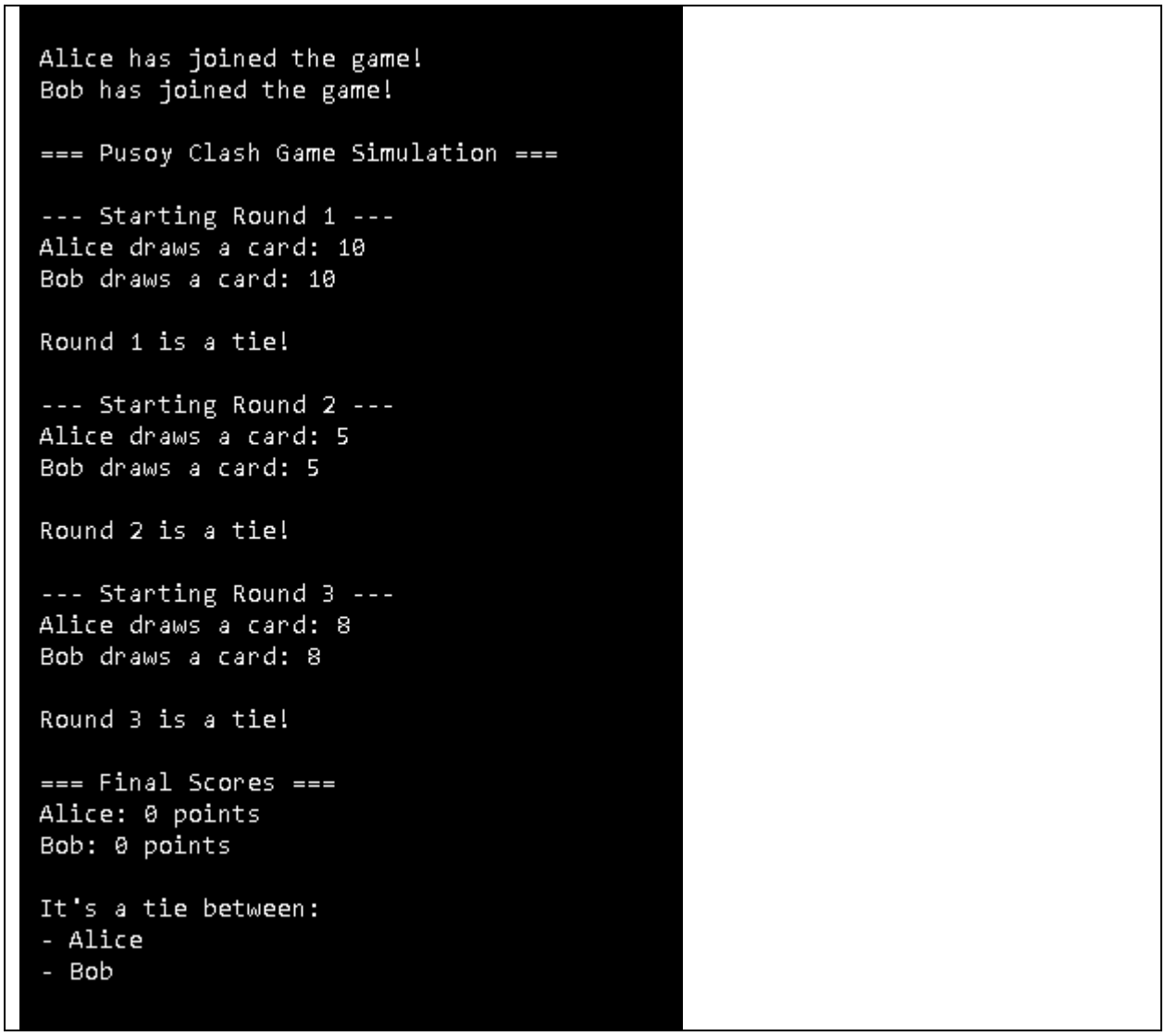
TEST CASES

Test Case 1: 1 Person Win

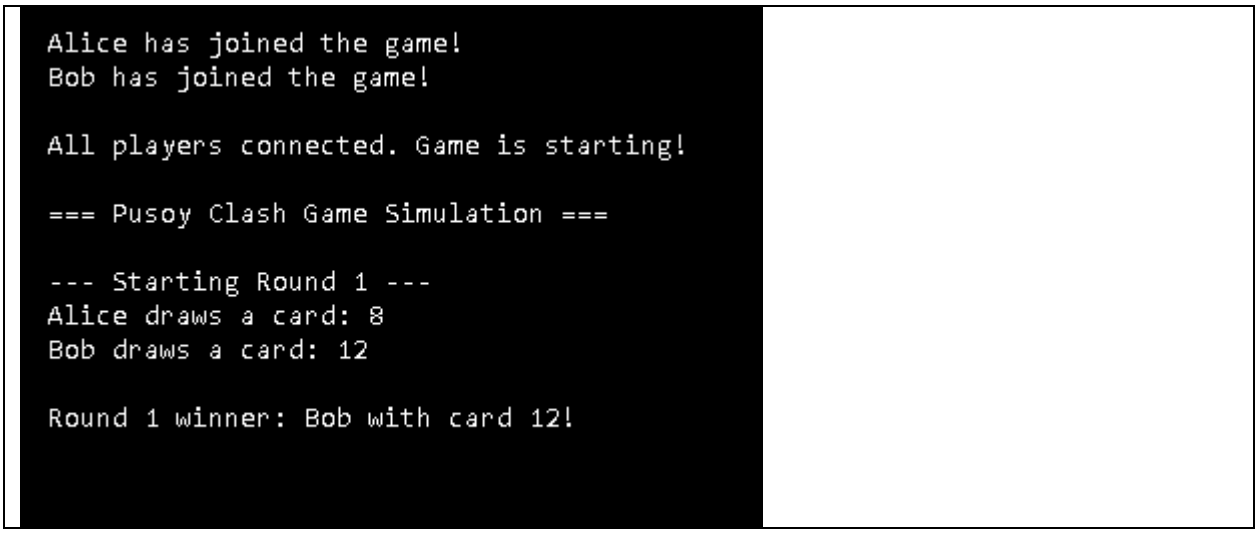
```
Waiting for players...
Alice has joined the game!
Bob has joined the game!

All players connected. Game is starting!

=== Pusoy Clash Game Simulation ===

--- Starting Round 1 ---
Alice draws a card: 9
Bob draws a card: 11

Round 1 winner: Bob with card 11!

--- Starting Round 2 ---
Alice draws a card: 10
Bob draws a card: 10

Round 2 is a tie!

--- Starting Round 3 ---
Alice draws a card: 12
Bob draws a card: 3

Round 3 winner: Alice with card 12!

=== Final Scores ===
Alice: 1 points
Bob: 1 points

It's a tie between:
- Alice
- Bob
```

Test Case 2: Server Disconnects

Test Case 3: Tie every round



Test Case 4: Other Player Disconnects


Alice has joined the game!
Bob has joined the game!
All players connected. Game is starting!
=== Pusoy Clash Game Simulation ===
--- Starting Round 1 ---
Alice draws a card: 8
Bob draws a card: 12
Round 1 winner: Bob with card 12!