```
////////////////////////////
//Stanley Lim 50441072
//computer networks class
// TCP program assignment 1
//
// this will relay messages between the client and server
// This one will send messages with some garbage, but
//it was outputting garbage with the text before I started working on it
//It should work for the most part, at least on my end
////////////////////////////

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#include <iostream>


#define PORT_FOR_CLIENT 11072
#define PORT_FOR_SEVER 11075
#define MESSAGESIZE 50
#define SERV_HOST_ADDR "147.97.156.237"

using std::cout;
using std::cerr;

int main( int argc, char*argv[]){

        int sockfd;
        int clientSockfd;
        int serverSockfd;
        int nread;
        int addrlen;
        int client_addrlen;
        int my_relay_port_id = PORT_FOR_CLIENT;
        int server_port_id = PORT_FOR_SEVER;
        char serv_host_addr[20]=SERV_HOST_ADDR;

        //struct defined from netinet
        struct sockaddr_in my_addr, client_addr;
        struct sockaddr_in server_addr, back_addr;
        char msg[MESSAGESIZE];

        if(argc == 2){
                my_relay_port_id = atoi(argv[1]);
        }

        //INITIALIZATION
        //system call to create new socket for client to connect to
        if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                cerr << "Relay: socket error: " << errno << "\n";
                exit(1);
        }
        memset( &my_addr, 0, sizeof(my_addr));    // Zero out structure
        my_addr.sin_family = AF_INET;        // Internet address family
        my_addr.sin_addr.s_addr = htonl(INADDR_ANY);    // Any incoming interface
        my_addr.sin_port = htons(my_relay_port_id);    // my port

        if((serverSockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                cerr << "Relay: socket error: " << errno << "\n";
                exit(1);
        }

        memset( &server_addr, 0, sizeof(server_addr));    // Zero out structure
        server_addr.sin_family = AF_INET;        // Internet address family
        server_addr.sin_port = htons(server_port_id);    // my port


        if (inet_aton(serv_host_addr, &(server_addr.sin_addr))==0) // get server addr
{  // invalid server address
   cerr << "Client: Invalid server address...\n";
   exit(2);
}

        //BINDING
        //System call that binds a socket to an address
        if((bind(sockfd, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0))
        {
        cerr << "Server: bind fail: " << errno << "\n";
        exit(2);
        }

        listen(sockfd, SOMAXCONN);  //  decide by OS
        cout << "\n\nWaiting for client's message to relay to server....\n\n";

        client_addrlen = sizeof(client_addr);

        if (connect(serverSockfd, (struct sockaddr *) &server_addr, sizeof(server_addr))<0)
 {
        cerr << "Client: connect failed: "<< errno <<"....\n";
        exit(3);
```

```
    }

	while(1){
		//create socket to communicate on
		clientSockfd = accept(sockfd, (struct sockaddr *) &client_addr, (socklen_t
*)&client_addrlen);
		if (clientSockfd < 0)
		{
			cerr << "Server: accept error: " << errno << "\n";
			exit(2);
		}

		cout << "Connection from client in relay(" << inet_ntoa(client_addr.sin_addr);
		cout << ":" << ntohs(client_addr.sin_port) << ")\n";

		//Client relay
		nread = read(clientSockfd, msg, MESSAGESIZE);
		if(nread >0){
			cout << "Client\'s message: ";
			cout.write(msg, nread);
			cout << "\n";
		}

		nread = write(serverSockfd, msg, strlen(msg));
		if(nread < 0){
			cerr << "server's send failed...\n";
			exit(3);
		}

		//server relay
		nread = read(serverSockfd, msg, MESSAGESIZE);
		if(nread >0){
			cout << "Client's message From Server: ";
			cout.write(msg, nread);
			cout << "\n";
		}

		nread = write(clientSockfd, msg, strlen(msg));
		if(nread < 0){
			cerr << "server's send failed...\n";
			exit(3);
		}
		close(clientSockfd);
		close(serverSockfd);
	}
	close(sockfd);
	return 0;
}
```