```c
// GROUP B
// Nathan Baker
// nathan.t.baker@okstate.edu

#include "header.h"

int connect_to_server(char* ip_addr, int port) {
    // standard procedure for connecting to a server via TCP socket.
    int sock = 0;
    struct sockaddr_in serv_addr;
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\nsocket creation error.\n");
        return -1; // error
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port);
    if(inet_pton(AF_INET, ip_addr, &serv_addr.sin_addr)<=0) {
        printf("\ninvalid address.\n");
        return -1; // error
    }
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        printf("\nconnection failed.\n");
        return -1; // error
    }
    return sock; // socket descriptor
}

int main(int argc, char const *argv[]) {
    int port = 0;
    if (argc > 1) port = atoi(argv[1]);
    if (port > 3 || port < 1) { // only servers 1, 2, 3 exist
        printf("no valid server index provided.\n");
        exit(1);
    }
    char ip_addr[15];
    strcpy(ip_addr,"127.0.0.1"); // LOCALHOST
    // strcpy(ip_addr,"10.203.72.25"); // CSX1

    int sock = connect_to_server(ip_addr,8000+port);
    if (sock == -1) exit(1);
    char m[1000]; // to hold messages sent and received

    read(sock, &m, sizeof(m)); // read initial message from server
    char a = m[0];
    memmove(m, m+1, 1000); // remove first character and store as flag
    printf("%s",m);
    if (a == '2') exit(0); // if flag is 2, server is too busy.

    read(sock, &m, sizeof(m)); // read message from server thread.
    a = m[0];
    memmove(m, m+1, 1000);  // remove first character and store as flag
    printf("%s",m);
    if (a == '2') exit(0);// if flag is 2, end connection.

    while (1) {
        strcpy(m,""); // clear message buffer for scanning.
        if (a != '1') { // if flag is not 1, server is expecting a response.
            while (strlen(m) == 0) { // ensure customer entered at least 1 character.
                scanf("%1000[^\n]",m); // scan until newline.
                char z;
                if (strlen(m) == 0) scanf("%c",&z);
            }
            while ((a = getchar()) != '\n' && a != EOF) { } // flush input steam.
            send(sock, &m, sizeof(m), 0); // send message.
        }

        read(sock, &m, sizeof(m));
        a = m[0];
        memmove(m, m+1, 1000); // read message and separate first character flag.
        printf("%s",m);
        if (a == '2') exit(0); // if flag is 2, end connection. otherwise continue.
    }

    return 0;
}
```