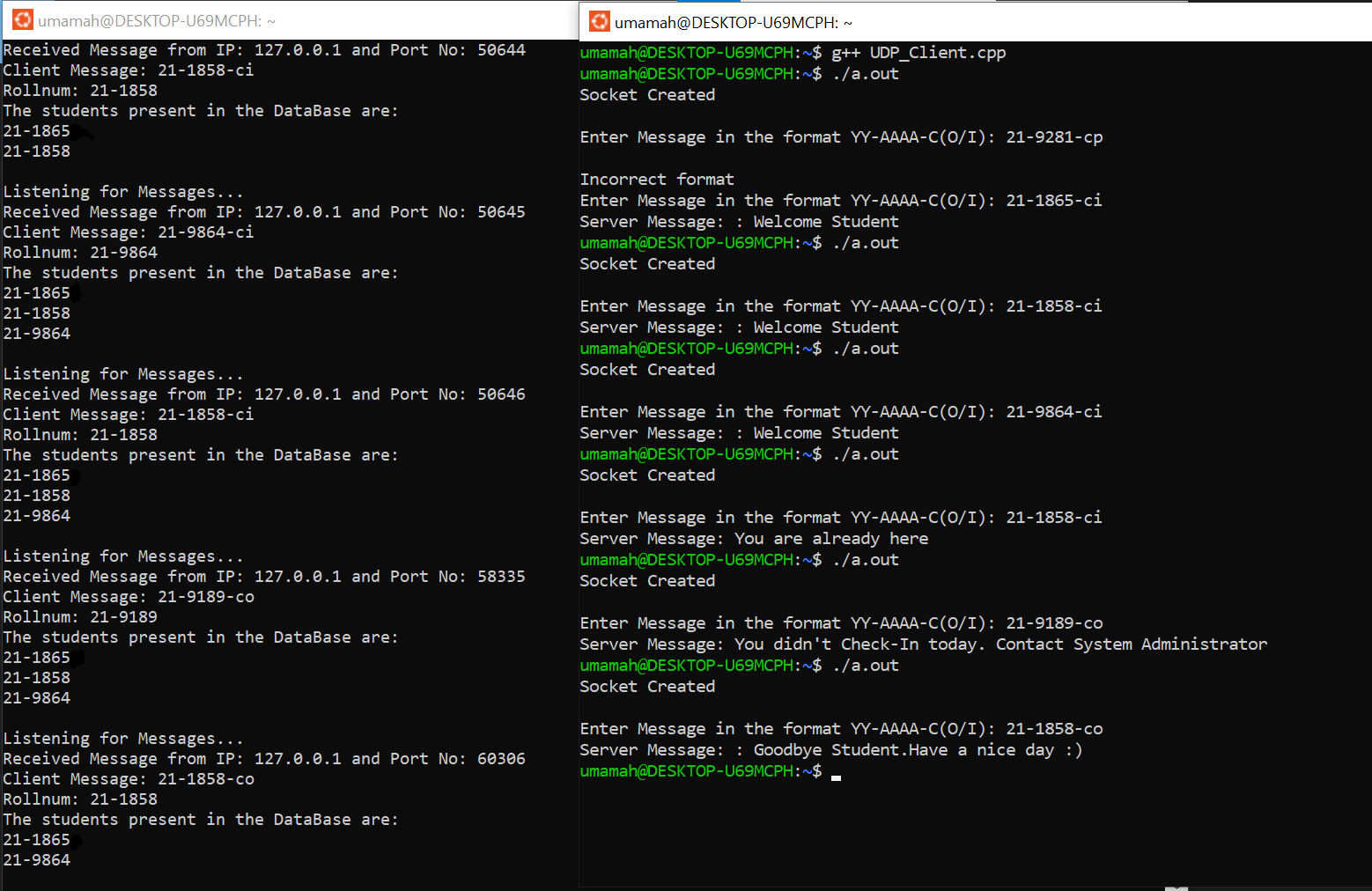
**Lab 6**

**Umamah Hussain**

**21L-1858**



**Server**

*/\**

*UDP\_Server. This Program will create the Server side for UDP\_Socket Programming.*

*It will receive the data from the client and then send the same data back to client.*

*\*/*

#include <stdio.h>

#include <unistd.h>

#include <iostream>

#include <algorithm>

#include <vector>

#include <string.h>

#include <sys/socket.h> *//// Needed for socket creating and binding*

#include <arpa/inet.h>  *//inet\_addr*

using namespace std;

int main(void)

{

        int socket\_desc;

        struct sockaddr\_in server\_addr, client\_addr;

        char server\_message[2000], client\_message[2000];

        socklen\_t client\_struct\_length = sizeof(client\_addr);

        int first = 0;

        vector<string> database;

        char msg[60];

*// Cleaning the Buffers*

        memset(server\_message, '\0', sizeof(server\_message));

        memset(client\_message, '\0', sizeof(client\_message));

*// Creating UDP Socket*

        socket\_desc = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

        if (socket\_desc < 0)

        {

                printf("Could Not Create Socket. Error!!!!!\n");

                return -1;

        }

        printf("Socket Created\n");

*// Binding IP and Port to socket*

        server\_addr.sin\_family = AF\_INET;

        server\_addr.sin\_port = htons(2000);

        server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); *// bind your socket to localhost only, if you want connect any particular ip you should mention it in INET\_ADDR.*

        if (bind(socket\_desc, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

        {

                printf("Bind Failed. Error!!!!!\n");

                return -1;

        }

        printf("Bind Done\n");

        while (1)

        {

                printf("\nListening for Messages...\n");

*// Receive the message from the client*

                if (recvfrom(socket\_desc, client\_message, sizeof(client\_message), 0, (struct sockaddr \*)&client\_addr, &client\_struct\_length) < 0)

                {

                        printf("Receive Failed. Error!!!!!\n");

                        return -1;

                }

                printf("Received Message from IP: %s and Port No: %i\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));

                printf("Client Message: %s\n", client\_message);

                char rollnum[7]={'0','0','0','0','0','0','0'};

                for (int i = 0; i <= 6; i++)

                {

                        rollnum[i] = client\_message[i];

                }

                printf("Rollnum: %s",rollnum);

                if (client\_message[9] == 'i' || client\_message[9] == 'I')

                {

*// printf("\nEntering the Check-In Mechanism\n");*

                        if (first == 0)

                        {

                                database.push\_back(rollnum);

                                first++;

*// printf("Welcome Student: %s",rollnum);*

                                strcat(msg,rollnum);

                                strcpy(msg,": Welcome Student");

                        }

                        else

                        {

                                auto it = find(database.begin(), database.end(), rollnum);

                                if (it != database.end())

                                {

                                       strcpy(msg,"You are already here");

                                }

                                else

                                {

                                        database.push\_back(rollnum);

*// printf("Welcome Student: %s",rollnum);*

                                       strcat(msg,rollnum);

                                       strcpy(msg,": Welcome Student");

                                }

                        }

                }

                else

                {

*//  printf("\nEntering the Check-Out Mechanism\n");*

                         if(first==0)

                         {

                               strcpy(msg,"No one has logged in yet");

                         }

                         else

                         {

                                auto it = find(database.begin(), database.end(), rollnum);

                                if (it != database.end())

                                {

                                        database.erase(it);

                                        strcat(msg,rollnum);

                                        strcpy(msg,": Goodbye Student.Have a nice day :)");

                                }

                                else

                                {

                                        strcpy(msg,"You didn't Check-In today. Contact System Administrator");

                                }

                         }

                }

                cout<<"\nThe students present in the DataBase are: \n";

                for(auto it=database.begin();it!=database.end();it++)

                {

                        cout<<\*it;

                        cout<<endl;

                }

        strcpy(server\_message, msg);

        if (sendto(socket\_desc, server\_message, strlen(server\_message), 0, (struct sockaddr\*)&client\_addr,client\_struct\_length)<0)

        {

                printf("Send Failed. Error!!!!!\n");

                return -1;

        }

        memset(server\_message, '\0', sizeof(server\_message));

        memset(msg, '\0', sizeof(msg));

        }

        close(socket\_desc);

        return 0;

}

**Client**

*/\**

*UDP\_Client. This Program will implement the Client Side for UDP\_Socket Programming.*

*It will get some data from user and will send to the server and as a reply from the*

*server, it will get its data back.*

*\*/*

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <sys/socket.h> *// Needed for socket creating and binding*

#include <arpa/inet.h>  *//inet\_addr*

int main(void)

{

        int socket\_desc;

        struct sockaddr\_in server\_addr;

        char server\_message[2000], client\_message[2000];

        socklen\_t server\_struct\_length = sizeof(server\_addr);

*// Cleaning the Buffers*

        memset(server\_message, '\0', sizeof(server\_message));

        memset(client\_message, '\0', sizeof(client\_message));

*// Creating UDP Socket*

        socket\_desc = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

        if (socket\_desc < 0)

        {

                printf("Could Not Create Socket. Error!!!!!\n");

                return -1;

        }

        printf("Socket Created\n");

*// Specifying the IP and Port of the server to connect*

        server\_addr.sin\_family = AF\_INET;

        server\_addr.sin\_port = htons(2000);

        server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); *// bind your socket to localhost only, if you want connect any particular ip you should mention it in INET\_ADDR.*

*// Get Input from the User*

        int input = 0;

        while (1)

        {

                if (input == 0)

                {

                        printf("\nEnter Message in the format YY-AAAA-C(O/I): ");

                        scanf("%s", client\_message);

                        input = 1;

                }

                if (input == 1)

                {

                        if (strlen(client\_message) != 10 &&

                        (client\_message[8] != 'C' && client\_message[8] != 'c') ||

                        (client\_message[9] != 'O' && client\_message[9] != 'o' &&

                        client\_message[9] != 'I' && client\_message[9] != 'i'))

                        {

                                printf("\nIncorrect format");

                                input = 0;

                        }

                        else

                                input = 2;

                }

                if (input == 2)

                {

                        int check = 0;

                        char arr[] = "0123456789";

                        for (int i = 0; i <= 9; i++)

                        {

                                if (i == 2 || i == 7 || i == 8 || i == 9)

                                        continue;

                                else

                                {

                                        check = 0;

                                        for (int j = 0; j < 11; j++)

                                        {

                                                if (client\_message[i] == arr[j])

                                                {

                                                        check = 1;

                                                }

                                        }

                                        if (check == 0)

                                        {

                                                printf("\nIncorrect format");

                                                input = 0;

                                                break;

                                        }

                                        else

                                                input = 3;

                                }

                        }

                }

                if (input == 3)

                {

                        break;

                }

        }

*// Send the message to Server*

        if (sendto(socket\_desc, client\_message, strlen(client\_message), 0, (struct sockaddr \*)&server\_addr, server\_struct\_length) < 0)

        {

                printf("Send Failed. Error!!!!\n");

                return -1;

        }

*// Receive the message back from the server*

        if (recvfrom(socket\_desc, server\_message, sizeof(server\_message), 0, (struct sockaddr \*)&server\_addr, &server\_struct\_length) < 0)

        {

                printf("Receive Failed. Error!!!!!\n");

                return -1;

        }

        printf("Server Message: %s\n", server\_message);

        memset(server\_message, '\0', sizeof(server\_message));

        memset(client\_message, '\0', sizeof(client\_message));

*// Closing the Socket*

        close(socket\_desc);

        return 0;

}