```
// ***** tcpclient.cpp *****

//**********************************************************************
//
// Computer Science 4/5313 Computer Networks
//
// Spring  2016
//
// Instructor:  Hung-Chi Su
//
// Assignment # n
//
// Programmer:  your name
//
// Due Date:    day-of-week, month day, year
//
// Description: This is a TCP socket program that illustrates how a client
//              connect to TCP server (tcpserver.cpp) and send/receive a
//              message to/from the server
//
// Editor/Platform:  vi/Linux
//
//
//
//  Input:      none
//
//  Output:     The server will display the message sent by this client
//
//  Compile:    g++ -o tcpclient tcpclient.cpp
//
//  Command:    (After server is running)
//              ./tcpclient  [<port#>  [<IP>]]
//
//
//  Note:       Remember to Change SERVER_PORT_ID to be same as server's
//
//**********************************************************************

                              // header files

#include <stdio.h>         //for printf(), ...
#include <stdlib.h>
#include <string.h>

//#include <sys/types.h>   //for data types
#include <sys/socket.h>    //for socket(), connect(), ...
#include <unistd.h>        //for close()
#include <netinet/in.h>    //for internet address family
#include <arpa/inet.h>     //for sockaddr_in, inet_addr()
```

```
#include <errno.h>          //for system error numbers
#include <iostream>         //for cin, cout...

// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// change the last 4 digits to your last 4 digits of student ID
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#define SERVER_PORT_ID  11072      //or 21234, 31234,41234,51234

#define SERV_HOST_ADDR "147.97.156.237"
//#define SERV_HOST_ADDR "127.0.0.1"

#define MESSAGESIZE        80

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

main(int argc, char *argv[])
{
  int sockfd,     // socket descriptor
      retcode;    // return code form some functions
  int server_port_id = SERVER_PORT_ID;
  char serv_host_addr[20]=SERV_HOST_ADDR;
  struct sockaddr_in server_addr;  // addresses for sockets
  char msg[MESSAGESIZE];     // buffer

  if (argc>1)
  {
     server_port_id = atoi(argv[1]);

     if (argc == 3)
     {
        if (strlen(argv[2])>=20)
        {
           cerr << "Too long address!!";
           exit(1);
        }
        strcpy(serv_host_addr, argv[2]);
     }
  }
```

```
// -------------------------------------------------------
// Initialization:
// -------------------------------------------------------
//cout << "Client: creating socket\n";
//if ( (sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
   cerr << "Client: socket failed: "<< errno <<"....\n";
   exit(1);
}

// cout << "Client: constructing addr structure for server\n";
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);// server port
//server_addr.sin_addr.s_addr = inet_addr(serv_host_addr); // old style
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);
}

// -------------------------------------------------------
// Establish the connection
// -------------------------------------------------------
if (connect(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr))<0)
{
   cerr << "Client: connect failed: "<< errno <<"....\n";
   exit(3);
}

// -------------------------------------------------------
// Message Preparation
// -------------------------------------------------------
// cout << "Client: initializing message and sending\n";
// memset(msg, 0, MESSAGESIZE);
//strcpy(msg,  "Hello world..............");
cout << "Please type a short message\n";
cin.getline(msg, MESSAGESIZE,'\n');
```

```
  // ---------------------------------------------------------
  // Transmission
  // ---------------------------------------------------------
  retcode = write(sockfd,msg,strlen(msg));
  if (retcode < 0)
  {
     cerr << "client: send failed: " << errno << "....\n";
     exit(4);
  }

  retcode = read(sockfd,msg, MESSAGESIZE);
  // cout << "client: retrun code from read is " << nread << "\n";
  if (retcode >0)
  {
    cout << "Relay's message is: ";
    cout.write(msg, retcode);                    // Message from server
    // cout << msg ;  // incorrect way to do
    cout << "\n";
  }
  // ---------------------------------------------------------
  // Termination
  // ---------------------------------------------------------
  /* close socket */
  close(sockfd);
}
```