```
//==================== file = udpClient.c =====Pseudo Code===============
//=  A message "client" program to demonstrate sockets programming
=
//=============================================================================
==
//=  Notes:
=
//=    1) This program conditionally compiles for Winsock and BSD sockets.
=
//=       Set the initial #define to WIN or BSD as appropriate.
=
//=    2) This program needs udpServer to be running on another host.
=
//=       Program udpServer must be started first.
=
//=    3) This program assumes that the IP address of the host running
=
//=       udpServer is defined in "#define IP_ADDR"
=
//=----------------------------------------------------------------------------
-=
//=  Example execution: (udpServer and udpClient running on host
computerassignmenthelp )   =
//=    Received from server: This is a reply message from SERVER to CLIENT
=
//=----------------------------------------------------------------------------
-=
//=  Build: bcc32 ucpClient.c or cl udpClient.c wsock32.lib for Winsock
=
//=         gcc udpClient.c -lsocket -lnsl for BSD
=
//=----------------------------------------------------------------------------
-=
//=  Execute: udpClient
=
//=============================================================================
==
#define  WIN                // WIN for Winsock and BSD for BSD sockets

//----- Include files ---------------------------------------------------------
--
#include <stdio.h>          // Needed for printf()
#include <string.h>         // Needed for memcpy() and strcpy()
#include <stdlib.h>         // Needed for exit()
#ifdef WIN
  #include <windows.h>      // Needed for all Winsock
#endif
#ifdef BSD
  #include <sys/types.h>    // Needed for sockets
  #include <netinet/in.h>   // Needed for sockets
  #include <sys/socket.h>   // Needed for sockets
  #include <arpa/inet.h>    // Needed for sockets
  #include <fcntl.h>        // Needed for sockets
  #include <netdb.h>        // Needed for sockets
#endif
```

```
//----- Defines ------------------------------------------------------------
--
#define  PORT_NUM           1050  // Port number used
#define  IP_ADDR      "127.0.0.1" // IP address of server1 (*** HARDWIRED
***)

//===== Main program
==========================================================
int main()
{
#ifdef WIN
  WORD wVersionRequested = MAKEWORD(1,1);       //  for WSA functions
  WSADATA wsaData;                              //  for WSA functions
#endif
  int                  client_s;        // Client socket descriptor
  struct sockaddr_in   server_addr;     // Server Internet address
  int                  addr_len;        // Internet address length
  char                 out_buf[4096];   // Output buffer for data
  char                 in_buf[4096];    // Input buffer for data
  int                  retcode;         // Return code

#ifdef WIN
  // This  initializes winsock
  WSAStartup(wVersionRequested, &wsaData);
#endif

  // >>> Step #1 <<<
  // Create a socket
  //   - AF_INET is Address Family Internet and SOCK_DGRAM is datagram
  client_s = socket(AF_INET, SOCK_DGRAM, 0);
  if (client_s < 0)
  {
    printf("*** ERROR - socket() failed \n");
    exit(-1);
  }

  // >>> Step #2 <<<
  // Fill-in server1 socket's address information
  server_addr.sin_family = AF_INET;                 // Address family to use
  server_addr.sin_port = htons(PORT_NUM);           // Port num to use
  server_addr.sin_addr.s_addr = inet_addr(IP_ADDR); // IP address to use

  // Assign a message to buffer out_buf
  strcpy(out_buf, "Test message from CLIENT to SERVER");

  // >>> Step #3 <<<
  // Now send the message to server.  The "+ 1" is for the end-of-string
  // delimiter
  retcode = sendto(client_s, out_buf, (strlen(out_buf) + 1), 0,
    (struct sockaddr *)&server_addr, sizeof(server_addr));
  if (retcode < 0)
  {
    printf("*** ERROR - sendto() failed \n");
    exit(-1);
  }

  // >>> Step #4 <<<
```

```
  // Wait to receive a message
  addr_len = sizeof(server_addr);
  retcode = recvfrom(client_s, in_buf, sizeof(in_buf), 0,
    (struct sockaddr *)&server_addr, &addr_len);
  if (retcode < 0)
  {
    printf("*** ERROR - recvfrom() failed \n");
    exit(-1);
  }

  // Output the received message
  printf("Received from server: %s \n", in_buf);

  // >>> Step #5 <<<
  // Close all open sockets
#ifdef WIN
  retcode = closesocket(client_s);
  if (retcode < 0)
  {
    printf("*** ERROR - closesocket() failed \n");
    exit(-1);
  }
#endif
#ifdef BSD
  retcode = close(client_s);
  if (retcode < 0)
  {
    printf("*** ERROR - close() failed \n");
    exit(-1);
  }
#endif

#ifdef WIN
  // This  cleans-up winsock
  WSACleanup();
#endif

  // Return zero and terminate
  return(0);
}
```