```
//====================file = tcpServer.c Pseudo code =======================
//=  A message "server" 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 serves a message to program tcpClient running on
=
//=       another host.
=
//=---------------------------------------------------------------------------
-=
//=  Example execution: (tcpServer and tcpClient running on host)   =
//=    Waiting for accept() to complete...
=
//=    Accept completed      =
//=    Received from client: This is a reply message from CLIENT to SERVER
=
//=---------------------------------------------------------------------------
-=
//=  Build: bcc32 tcpServer.c or cl tcpServer.c wsock32.lib for Winsock
=
//=         gcc tcpServer.c -lnsl for BSD
=
//=---------------------------------------------------------------------------
-=
//=  Execute: tcpServer
=
//=-------------------------------------------------------------------------
//=============Pseudo Code=================================================


#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    // Arbitrary port number for the server

//===== Pseudo program
============================================================
int main()
{
#ifdef WIN
  WORD wVersionRequested = MAKEWORD(1,1);       //  for WSA functions
  WSADATA wsaData;                              //  for WSA functions
#endif
  int                 welcome_s;       // Welcome socket descriptor
  struct sockaddr_in  server_addr;     // Server Internet address
  int                 connect_s;       // Connection socket descriptor
  struct sockaddr_in  client_addr;     // Client Internet address
  struct in_addr      client_ip_addr;  // Client IP 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 welcome socket
  //   - AF_INET is Address Family Internet and SOCK_STREAM is streams
  welcome_s = socket(AF_INET, SOCK_STREAM, 0);
  if (welcome_s < 0)
  {
    printf("*** ERROR - socket() failed \n");
    exit(-1);
  }

  // >>> Step #2 <<<
  // Fill-in server (my) address information and bind the welcome socket
  server_addr.sin_family = AF_INET;                // Address family to use
  server_addr.sin_port = htons(PORT_NUM);          // Port number to use
  server_addr.sin_addr.s_addr = htonl(INADDR_ANY);  // Listen on any IP
address
  retcode = bind(welcome_s, (struct sockaddr *)&server_addr,
    sizeof(server_addr));
  if (retcode < 0)
  {
    printf("*** ERROR - bind() failed \n");
    exit(-1);
  }

  // >>> Step #3 <<<
  // Listen on welcome socket for a connection
  listen(welcome_s, 1);

  // >>> Step #4 <<<
  // Accept a connection.  The accept() will block and then return with
```

```
  // connect_s assigned and client_addr filled-in.
  printf("Waiting for accept() to complete... \n");
  addr_len = sizeof(client_addr);
  connect_s = accept(welcome_s, (struct sockaddr *)&client_addr, &addr_len);
  if (connect_s < 0)
  {
    printf("*** ERROR - accept() failed \n");
    exit(-1);
  }

  // Copy the four-byte client IP address into an IP address structure
  memcpy(&client_ip_addr, &client_addr.sin_addr.s_addr, 4);

  // Print an informational message that accept completed
  printf("Accept completed (IP address of client = %s  port = %d) \n",
    inet_ntoa(client_ip_addr), ntohs(client_addr.sin_port));

  // >>> Step #5 <<<
  // Send to the client using the connect socket
  strcpy(out_buf, "This is a message from SERVER to CLIENT");
  retcode = send(connect_s, out_buf, (strlen(out_buf) + 1), 0);
  if (retcode < 0)
  {
    printf("*** ERROR - send() failed \n");
    exit(-1);
  }

  // >>> Step #6 <<<
  // Receive from the client using the connect socket
  retcode = recv(connect_s, in_buf, sizeof(in_buf), 0);
  if (retcode < 0)
  {
    printf("*** ERROR - recv() failed \n");
    exit(-1);
  }
  printf("Received from client: %s \n", in_buf);

  // >>> Step #7 <<<
  // Close the welcome and connect sockets
#ifdef WIN
  retcode = closesocket(welcome_s);
  if (retcode < 0)
  {
    printf("*** ERROR - closesocket() failed \n");
    exit(-1);
  }
  retcode = closesocket(connect_s);
  if (retcode < 0)
  {
    printf("*** ERROR - closesocket() failed \n");
    exit(-1);
  }
#endif
#ifdef BSD
  retcode = close(welcome_s);
  if (retcode < 0)
  {
```

```
    printf("*** ERROR - close() failed \n");
    exit(-1);
  }
  retcode = close(connect_s);
  if (retcode < 0)
  {
    printf("*** ERROR - close() failed \n");
    exit(-1);
  }
#endif

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

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