TCPEchoClient.c
From TCP/IP Sockets in C: Practical Guide for Programmers, Donohoo and Calvert.

```
#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/socket.h> /* for socket(), connect(), send(), and recv() */
#include <arpa/inet.h>  /* for sockaddr_in and inet_addr() */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */

#define RCVBUFSIZE 32   /* Size of receive buffer */

void DieWithError(char *errorMessage);  /* Error handling function */

int main(int argc, char *argv[])
{
    int sock;                        /* Socket descriptor */
    struct sockaddr_in echoServAddr; /* Echo server address */
    unsigned short echoServPort;     /* Echo server port */
    char *servIP;                    /* Server IP address (dotted quad) */
    char *echoString;                /* String to send to echo server */
    char echoBuffer[RCVBUFSIZE];     /* Buffer for echo string */
    unsigned int echoStringLen;      /* Length of string to echo */
    int bytesRcvd, totalBytesRcvd;   /* Bytes read in single recv()
                                        and total bytes read */

    if ((argc < 3) || (argc > 4))    /* Test for correct number of arguments */
    {
       fprintf(stderr, "Usage: %s <Server IP> <Echo Word> [<Echo Port>]\n",
               argv[0]);
       exit(1);
    }

    servIP = argv[1];             /* First arg: server IP address (dotted quad) */
    echoString = argv[2];         /* Second arg: string to echo */

    if (argc == 4)
        echoServPort = atoi(argv[3]); /* Use given port, if any */
    else
        echoServPort = 7;  /* 7 is the well-known port for the echo service */

    /* Create a reliable, stream socket using TCP */
    if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
        DieWithError("socket() failed");

    /* Construct the server address structure */
    memset(&echoServAddr, 0, sizeof(echoServAddr));     /* Zero out structure */
    echoServAddr.sin_family      = AF_INET;             /* Internet address family */
    echoServAddr.sin_addr.s_addr = inet_addr(servIP);   /* Server IP address */
    echoServAddr.sin_port        = htons(echoServPort); /* Server port */

    /* Establish the connection to the echo server */
    if (connect(sock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
        DieWithError("connect() failed");

    echoStringLen = strlen(echoString);          /* Determine input length */
```

```
    /* Send the string to the server */
    if (send(sock, echoString, echoStringLen, 0) != echoStringLen)
        DieWithError("send() sent a different number of bytes than expected");

    /* Receive the same string back from the server */
    totalBytesRcvd = 0;
    printf("Received: ");                /* Setup to print the echoed string */
    while (totalBytesRcvd < echoStringLen)
    {
        /* Receive up to the buffer size (minus 1 to leave space for
           a null terminator) bytes from the sender */
        if ((bytesRcvd = recv(sock, echoBuffer, RCVBUFSIZE - 1, 0)) <= 0)
            DieWithError("recv() failed or connection closed prematurely");
        totalBytesRcvd += bytesRcvd;   /* Keep tally of total bytes */
        echoBuffer[bytesRcvd] = '\0';  /* Terminate the string! */
        printf(echoBuffer);            /* Print the echo buffer */
    }

    printf("\n");    /* Print a final linefeed */

    close(sock);
    exit(0);
}
```

ResolveName.c
From TCP/IP Sockets in C: Practical Guide for Programmers, Donohoo and Calvert.

```
#include <stdio.h>      /* for fprintf() */
#include <netdb.h>      /* for gethostbyname() */
#include <stdlib.h>     /* for exit() */

unsigned long ResolveName(char name[])
{
    struct hostent *host;            /* Structure containing host information */

    if ((host = gethostbyname(name)) == NULL)
    {
        fprintf(stderr, "gethostbyname() failed");
        exit(1);
    }

    /* Return the binary, network byte ordered address */
    return *((unsigned long *) host->h_addr_list[0]);
}
```