```
//*********************************************************//
//                                                        //
//                    PROJECT: DNS DOSSIER                    //
//                                                        //
//*********************************************************//
//                                                        //
//                 CLIENT END TCP/IP APPLICATION               //
//                                                        //
//*********************************************************//

#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()
#include <stdbool.h>    // for ip check
#include <ctype.h>      // for tolower()


/* Function to validate the IP Address entered by Client */
bool isValidIpAddress(char *ipAddress)
 {
 struct sockaddr_in sa;
 int result = inet_pton(AF_INET, ipAddress, &(sa.sin_addr));
 return result != 0;
 }


#define RCVBUFSIZE 100             // Size of receive buffer

//****************    Function Prototype Declarations ***************//

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

char * toString(char str[], int num);

//-----------------------------------------------------------------//
//***************       MAIN FUNCTION       ************************//
//-----------------------------------------------------------------//

int main(int argc, char *argv[])
{
    int sock;                          // Socket descriptor
    struct sockaddr_in serverAddr;      // Echo server address
    unsigned short serverPort;          // Echo server port
    char *servIP;                      // Server IP address (dotted quad)
    char echoString[100];              // String to send to echo server
    char echoBuffer[RCVBUFSIZE];        // Buffer for echo string
```

```
unsigned long echoStringLen;        // Length of string to echo
long bytesRcvd, totalBytesRcvd;     // Bytes read in single recv() and total bytes read
char * action;                      // Type of request from client to server
char * domainName;                  // For Domain name argument
char * ipToAdd;
char str[2];                        //= argc;

if ((argc < 4) || (argc > 6))       // Test for correct number of arguments
{
   printf("\n\t\tNo of command line parametes aren't enough and proper for the request");
   exit(1);
}

servIP = argv[1];                   // First arg: server IP address (dotted quad)
serverPort = atoi(argv[2]);         // Use given port, if any
action = argv[3];

strcpy(echoString,toString(str,argc));
strcat(echoString, "#");            // Formatting the string to be sent with "#" in between args
strcat(echoString, action);
strcat(echoString, "#");            // Check if valid action code is entered

if (atoi(action)> 6 || atoi(action) <0)
   DieWithError("Invalid request code entered by the client");

switch (argc){                      // Check for the number of args entered by the Client
   case 5:    domainName = argv[4];
            strcat(echoString,domainName);
            strcat(echoString,"#");
            printf("\nCommand Sent: %s %s %s %s",argv[1],argv[2],argv[3],argv[4] );
            break;

   case 6:    //To validate the IP Address format
            if (isValidIpAddress(argv[5])){
               domainName = argv[4];
               strcat(echoString,domainName);  // Concatenate the domain name to the string
               strcat(echoString," ");
               ipToAdd = argv[5];
               strcat(echoString,ipToAdd);     // Concatenate the IP to the string
               strcat(echoString,"#");
               printf("\nCommand Sent: %s %s %s %s %s",argv[1],argv[2],argv[3],argv[4], argv[5]);
               break;
            }
            else
               DieWithError("Invalid IP Address entered by the client");

   default:   printf("\nCommand Sent: %s %s %s",argv[1],argv[2],argv[3]);
            break;
}

/* 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(&serverAddr, 0, sizeof(serverAddr));         // Zero out structure
serverAddr.sin_family      = AF_INET;               // Internet address family
serverAddr.sin_addr.s_addr = inet_addr(servIP);     // Server IP address
serverAddr.sin_port        = htons(serverPort);     // Server port

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

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

/* Converting all arguments to lowercase */
if (strcmp(action,"6") != 0 ){
    for(int  i = 0; echoString[i]; i++){
        if(echoString[i]!= '#')
            echoString[i] = tolower(echoString[i]);
        }
}

/* 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("\nReceived: ");                             // Setup to print the echoed string */

/* Receive up to the buffer size (minus 1 to leave space for
 a null terminator) bytes from the sender */
while (totalBytesRcvd < RCVBUFSIZE)
{
    /* 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("%s", echoBuffer);                       // Print the echo buffer
}

printf("%s", echoBuffer);                           // Print the echo buffer
```

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

   close(sock);                           // CLOSE SOCKET
   exit(0);
}
//----------------------------------------------------------------------//
//*********************     MAIN FUNCTION ENDS     *********************//
//----------------------------------------------------------------------//

// Die with error - Error handling function
void DieWithError(char *errorMessage)
{
   perror(errorMessage);
   exit(1);
}

//toString function
char * toString(char * str, int num)
{
   int i, rem, len = 0, n;

   n = num;
   while (n != 0)          // Iterates over to count the number of digits
   {
      len++;
      n /= 10;
   }
   for (i = 0; i < len; i++)   // Converts each digit into a character in the string
   {
      rem = num % 10;
      num = num / 10;
      str[len - (i + 1)] = rem +'0' ;
   }
   str[len] = '\0';

   return str;             // return the integer converted to String
}
```