```
/*-------client.c-------
sample code */

// Standard libraries
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Socket-related libraries
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

#define MAX_LINE_SIZE 50

void print_usage(){
  fprintf(stderr, "Usage: ./client [hostname] [port]\n");
  fprintf(stderr, "Example: ./client 192.168.0.2 4845\n\n");
}

void check_args(int argc, char *argv[]){
  if( argc != 3 ) {
    print_usage();
        exit(1);
  }
}

int main(int argc, char *argv[])
{

  struct addrinfo hints;
  struct addrinfo *serverInfo;

  // Check arguments and exit if they are not well constructed
  check_args(argc, argv);

  // Fill the 'hints' structure
  memset( &hints, 0, sizeof(hints) );
  hints.ai_family   = AF_INET;     // Use IPv4 address
        hints.ai_socktype = SOCK_STREAM; // TCP sockets

  // Get the address information of the server.
  // argv[1] is the address, argv[2] is the port
  getaddrinfo(argv[1], argv[2], &hints, &serverInfo);

  // Create a socket to communicate with the server
  int socketFD = socket( serverInfo->ai_family, serverInfo->ai_socktype,
serverInfo->ai_protocol );

  // Connect to the server (only for TCP)
  connect(socketFD, serverInfo->ai_addr, serverInfo->ai_addrlen);

  // Ask for data to send
  char line[MAX_LINE_SIZE];
  printf("Please input the string to send: ");
  scanf("%s", line);
  int len = strlen(line);
```

```
  // Send the data:
  printf("Sending string '%s' to '%s' using port '%s'... ", line, argv[1],
argv[2]);
  send(socketFD, line, len, 0); // TCP style. Use sendto() for UDP
  printf("Done!\n\n");

  // Free the address info struct and close the socket
  freeaddrinfo(serverInfo);
  close(socketFD);

  return EXIT_SUCCESS;
}
```