```c
/* CS480, Spring 2016
 *
 *   Simple UDP client, to demonstrate use of the sockets API
 *   Compile with:
 *        gcc -Wall -o udp-client udp-client.c
 *   or
 *        gcc -g -Wall -o udp-client udp-client.c  # to support debugging
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <assert.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <limits.h>

void handle_error(const char* s)
{
    perror(s);
    exit(1);
}

int main(int argc, char** argv)
{
    int sock_fd;
    struct sockaddr_in addr;
    struct hostent* hostentp;
    char* endptr;
    unsigned int portl;
    unsigned short port;
    size_t num_to_send;
    size_t num_sent;

    if (argc != 4) {
        fprintf(stderr, "usage: %s <hostname> <port> <message>\n", argv[0]);
        exit(1);
    }

    /* convert from string to int */
    portl = strtol(argv[2], &endptr, 10);
    if (endptr == NULL || portl == 0)
        handle_error("strtol");

    assert(portl < USHRT_MAX);
    port = (unsigned short)portl;

    /*
     * Below, we use the C idiom for "assign to a variable
     * and then check its value"
     */
    if (!(hostentp = gethostbyname(argv[1]))) {
        herror("gethostbyname");
        exit(1);
    }

    /*
     * Note! This code is not the same as what you need to do in lab1.
     */
    if ((sock_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        handle_error("socket");

    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    memcpy(&addr.sin_addr.s_addr,
           hostentp->h_addr_list[0],
           sizeof(struct in_addr));
    addr.sin_port = htons(port);

    printf("I am about to send %s to IP address %s and port %d\n",
            argv[3], inet_ntoa(addr.sin_addr), atoi(argv[2]));
```



```c

    num_to_send = strlen(argv[3]);

    num_sent = sendto(sock_fd,     /* socket */
                      argv[3],     /* buffer to send */
                      num_to_send, /* number of bytes to send */
                      0,          /* flags=0: bare-bones use case*/
                      (const struct sockaddr*)&addr, /* the destination */
                      sizeof(addr));  /* size of the destination struct */

    /*
     *  this is for education/demo purposes (there's a simpler way to
     *  write this error-checking code.) sendto() "commits" to sending
     *  the whole packet or nothing, so if less than the whole thing was
     *  sent, it better have been because there was an error (indicated
     *  by returning < 0).
     */
    if (num_sent != num_to_send) {
        assert(num_sent < 0);
        handle_error("sendto");
    }

    printf("I just sent the bytes!\n");

    exit(0);
}
```

```c
/* CS480, Spring 2016
 *
 *   Simple UDP server, to demonstrate use of the sockets API
 *   Compile with:
 *        gcc -Wall -o udp-server udp-server.c
 *   or
 *        gcc -g -Wall -o udp-server udp-server.c   # for debugging support
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <assert.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <limits.h>

void handle_error(const char* s)
{
    perror(s);
    exit(1);
}

int main(int argc, char** argv)
{
    int sock_fd;
    struct sockaddr_in my_addr, my_peer_addr;
    char* endptr;
    unsigned int portl;
    unsigned short port;
    int num_received;
    char msg[1024];
    socklen_t addrlen;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <port>\n", argv[0]);
        exit(1);
    }

    /* convert from string to int */
    portl = strtol(argv[1], &endptr, 10);
    if (endptr == NULL || portl == 0)
        handle_error("strtol");

    assert(portl < USHRT_MAX);
    port = (unsigned short)portl;

    /*
     * Note! This code is not the same as what you need to do in lab1.
     */
    if ((sock_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        handle_error("socket");

    memset(&my_addr, 0, sizeof(my_addr));
    my_addr.sin_family = AF_INET;
    my_addr.sin_addr.s_addr = INADDR_ANY;
    my_addr.sin_port = htons(port);

    if (bind(sock_fd,
             (struct sockaddr*)&my_addr,
             sizeof(struct sockaddr_in)) < 0)
        handle_error("bind");

    while (1) {

        addrlen = sizeof(struct sockaddr_in);

        if ((num_received = recvfrom(sock_fd,     /* socket */
                                     msg,         /* buffer */
                                     sizeof(msg),  /* size of buffer */
                                     0,              /* flags = 0 */
                                     /* who's sending */
                                     (struct sockaddr*)&my_peer_addr,
                                     /* length of buffer to receive peer info */
                                     &addrlen)) < 0)

                handle_error("recvfrom");

        assert(addrlen == sizeof(struct sockaddr_in));

        printf("I got message %.*s from host %s, src port %d\n",
               num_received, msg,
               inet_ntoa(my_peer_addr.sin_addr),
               ntohs(my_peer_addr.sin_port));


    }

    exit(0);
}
```