Socket Programming Instruction Sheet

In this lab, our goal is to make two machines talk to each other. You will have to write two programs: ***listener.c*** and ***talker.c*** based on the concepts you have learned so far.

listener essentially sits on a machine waiting for incoming messages on a specific port (say 5000). It prints whatever message it gets on that port onto the screen. **talker** reads whatever the user types on the command line and sends this message to the **listener**.

To make things easy for you, we have provided the skeleton of the program. Your job is to fill in the blanks. The provided data sheet contains the syntax of all the functions you would need for this exercise. Note that the variables used in the syntax may be very different from those used in the skeleton, so use caution.

Initially, you will run both **listener** and **talker** on the same machine utilizing the loop back feature. This will help in debugging the programs. For this, you will need to specify the destination address (listener's) as 127.0.0.1. The source address (talker's) can be obtained by typing "ifconfig eth0" at the command prompt. Once the programs are running correctly, you can pair with one of the other teams and test that your **talker** runs with their **listener** and vice-versa. In this case change the destination IP address appropriately.

```
/********  listener.c – listens for incoming messages on port 5000 ********/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MYPORT __________   // the port talker will be connecting to
#define MAXBUFLEN 100

int main(void) {
        int sockfd;
        struct sockaddr_in my_addr;    // my address information
        struct sockaddr_in their_addr; // talker's address information
        int addr_len, numbytes;
        char buf[MAXBUFLEN];

        if ((sockfd = socket(________, ________, ________)) == -1) {//exit on error
           perror("socket");
           exit(1);
        }

        my_addr.sin_family = __________;
        my_addr.sin_port = htons(________);
        my_addr.sin_addr.s_addr = ______________; // automatically fill with my IP
        memset(&(my_addr.sin_zero), 0, 8); // zero the rest of the struct
        if (bind(___________, _________,___________) == -1) {//exit on error
           perror("bind");
           exit(1);
        }

        addr_len = _____________;
        if ((numbytes=recvfrom(________,________,___________, ________,
                     ____________, __________)) == -1) {//exit on error
           perror("recvfrom");
           exit(1);
        }

        printf("got packet from %s IP address\n",inet_ntoa(______________));
        printf("packet is %d bytes long\n",___________);
        buf[numbytes] = '\0';
        printf("packet contains the message \"%s\"\n",_________);

        close(_________);
        return 0;
}

/***** talker.c reads input from command line and sends message to listener ******/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define LPORT _________   // the port listener is waiting on

int main(int argc, char *argv[]) {
        int sockfd;
    struct sockaddr_in their_addr; // listener's address information
    int numbytes;

    if (argc != 3) {
        fprintf(stderr,"usage: talker hostname message\n");
        exit(1);
    }

    if ((sockfd = socket(_______, ______________, _________)) == -1) {
        perror("socket");
        exit(1);
    }

    their_addr.sin_family = __________;
    their_addr.sin_port = htons(____________);
    inet_aton(________, ________); // fill the IP address in their_addr.sin_addr
    memset(&(their_addr.sin_zero), '\0', 8); // zero the rest of the struct

    if ((numbytes=sendto(_______, _________, __________, 0,
         ______________, _______________)) == -1) { //exit on error
        perror("sendto");
        exit(1);
    }
    printf("sent %d bytes to %s IP address \n", _________, inet_ntoa(_________));
    close(_______);
    return 0;
}
```