* Skeleton Code for Lab4

(You can run with the code for how to communicate between server and client, But you should change many parts to complete LAB 4)

Client:

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<netinet/in.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/time.h>
#include<errno.h>
#include<arpa/inet.h>
#include<string.h>

#define RECEIVER_HOST "anaconda#.uml.edu" /* Server machine */

/* Declaring errno */
extern int errno;

/* Function for error */
void report_error(char *s)

{
  printf("sender: error in %s, errno = %d\n",s,errno);
  exit(1);
}

/* Giving 'size' of message dynamically as argument */
void main(int argc, char *argv[])
{
  int s,i;
  int BUFSIZE = atoi(argv[1]);
  char msg[BUFSIZE];
  char received[BUFSIZE];
  struct sockaddr_in sa= {0};
  int length = sizeof(sa);
  struct hostent *hp;

  printf("Enter the message to be sent: \n");
  scanf("%s",msg);

  /* FILL SOCKET ADDRESS*/
  if((hp = gethostbyname(RECEIVER_HOST))==NULL)
    report_error("gethostbyname");

  bcopy((char*)hp->h_addr, (char *)&sa.sin_addr, hp->h_length);
  sa.sin_family = hp->h_addrtype;
  sa.sin_port = htons(<last four digits of student ID> + 20000);  /* define port
number based on student ID*/

  /* Creating the socket and returns error if unsuccessfull */
  if((s=socket(AF_INET, SOCK_DGRAM, PF_UNSPEC))== -1)
    report_error("socket");
  printf("Socket= %d\n",s);

  /* Sending the message to server and returns error if unsuccesfull */
  if(sendto(s, msg, BUFSIZE, 0, (struct sockaddr *) &sa, length)== -1)
    report_error("sendto");

  /* Receives message from server and returns error if unsuccesfull */
  recvfrom(s, received, BUFSIZE, 0, (struct sockaddr *) &sa, &length);
  printf("%s\n",received);
  close(s);
}
```

Server:

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<netinet/in.h>
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<strings.h>
#include<string.h>

#define RECEIVER_HOST "anaconda#.uml.edu"   /* Server machine */

/* Declaring errno */
extern int errno;

/* Function for printing error */
void report_error(char *s)
{
  printf("receiver: error in%s, errno = %d\n", s, errno);
  exit(1);
}

/* Dynamically giving the 'size' of message as argument */
void main(int argc, char *argv[])
{
  int size=50;
  int s;
  char m[200]="Request received!";
  char response[size];
  char msg[size];
  struct sockaddr_in sa = {0}, r_sa = {0};
  int r_sa_l = sizeof(r_sa);
  int len;
  int backlog = 5;
  struct hostent *hp;
  socklen_t length;
  strcpy(response,m); /* Copying the string m into response as couldnot initialize
variable-sized array */

  /* Creating the socket and returns error if unsuccesfull */
  if((s= socket(AF_INET, SOCK_DGRAM, PF_UNSPEC)) == -1)
    report_error("socket");

  sa.sin_family = AF_INET;
  sa.sin_addr.s_addr=INADDR_ANY;
  sa.sin_port = htons(<last four digits of student ID> + 20000);  /* define port
number based on student ID*/

  /* Binding the socket and returns error if unsuccesfull */
  if(bind(s, (struct sockaddr *)&sa, sizeof(sa))== -1)
    report_error("bind");

  listen(s, 10);
  length = sizeof(r_sa);

  /* Receiving message from client and returns error if unsuccessfull */
  if((len = recvfrom(s, msg, size, 0, (struct sockaddr *)&r_sa, &r_sa_l))== -1)
    report_error("recvfrom");
   printf("%s\n",msg);

  /* Sending response to client */
  sendto(s,response,size,0,(struct sockaddr *)&r_sa,r_sa_l);
  close(s);
}
```