SERVER:

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

#define PORT 8080

typedef struct
{
    char s_ip[50];
    char s_mac[50];
    char d_ip[50];
    char d_mac[50];
    char data[50];
}arp_packet;

void print_arp_packet(arp_packet *p, int pdmac, int pdata)
{
    printf("%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  printf("|%s", p->d_mac);
    if(pdata == 1)  printf("|%s", p->data);
}

arp_packet *init_arp_packet()
{
    arp_packet *p = malloc(sizeof(arp_packet));
    printf("Enter the details of the packet to send: \n");
    printf("Enter Source IP: ");
    scanf("%s", p->s_ip);
    printf("Enter Source MAC: ");
    scanf("%s", p->s_mac);
    printf("Enter Destination IP: ");
    scanf("%s", p->d_ip);
    strcpy(p->d_mac, "\0");
    printf("Enter Data to be sent: ");
    scanf("%s", p->data);
    return p;
}

void pactostring(arp_packet *p, char *str, int pdmac, int pdata)
{
    sprintf(str, 150, "%s|%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  sprintf(str, 50, "|%s", p->d_mac);
    if(pdata == 1)  sprintf(str, 50, "|%s", p->data);
}

void stringtopac(arp_packet *p, char *str, int pdmac, int pdata)
{
    sscanf(str, 150, "%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  sscanf(str, 50, "|%s", p->d_mac);
    if(pdata == 1)  sscanf(str, 50, "|%s", p->data);
}

int main(int argc, char **argv)
{
    int len, sfd, cfd, n;
```

```
    struct sockaddr_in saddr, caddr;
    char buffer[1024], content[1024];
    pid_t child;

    //create a socket and get the socket fd
    sfd = socket(AF_INET, SOCK_STREAM, 0); //socket(int domain,int type,int
protocol)
    if(sfd<0)
        perror("Cannot create socket");

    //create server sockaddr and bind
    bzero(&saddr, sizeof(saddr));
    saddr.sin_family = AF_INET;
    saddr.sin_addr.s_addr = INADDR_ANY;
    saddr.sin_port = htons(PORT);

    if(bind(sfd, (struct sockaddr*)&saddr, sizeof(saddr)) < 0)
        perror("Bind Error");

    arp_packet *p;
    listen(sfd, 10);

    len = sizeof(caddr);
    printf("Server\n");
    p = init_arp_packet(p);
    printf("\nDeveloping ARP Request Packet\n");
    print_arp_packet(p, 0, 0);
    char *sent_pack;
    pactostring(p, buffer, 0, 0);
    pactostring(p, sent_pack, 0, 0);
    printf("\nThe ARP Request packet is broadcasted\nWaiting for ARP
reply...\n");

    while(1)
    {
        cfd = accept(sfd, (struct sockaddr *)&caddr, &len);
        if(cfd < 0) exit(1);
        if((child = fork()) == 0)
        {
            close(sfd);
            send(cfd, buffer, sizeof(buffer), 0);
            recv(cfd, buffer, sizeof(buffer), 0);
            while(strcmp(buffer, sent_pack) == 0)   recv(cfd, buffer,
sizeof(buffer), 0);
            stringtopac(p, buffer, 1, 0);
            printf("ARP reply recieved: ");
            print_arp_packet(p, 1, 0);
            printf("\nSending to: %s\n", p->d_mac);
            printf("Package Sent: ");
            print_arp_packet(p, 1, 1);
            pactostring(p, buffer, 1, 1);
            send(cfd, buffer, sizeof(buffer), 0);
            exit(1);
        }
        close(cfd);
    }

}
```

CLIENT:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>

#define PORT 8080

typedef struct
{
    char s_ip[50];
    char s_mac[50];
    char d_ip[50];
    char d_mac[50];
    char data[50];
}arp_packet;

void print_arp_packet(arp_packet *p, int pdmac, int pdata)
{
    printf("%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  printf("|%s", p->d_mac);
    if(pdata == 1)  printf("|%s", p->data);
}

void pactostring(arp_packet *p, char *str, int pdmac, int pdata)
{
    sprintf(str, 150, "%s|%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  sprintf(str, 50, "|%s", p->d_mac);
    if(pdata == 1)  sprintf(str, 50, "|%s", p->data);
}

void stringtopac(arp_packet *p, char *str, int pdmac, int pdata)
{
    sscanf(str, 150, "%s|%s|%s", p->s_ip, p->s_mac, p->d_ip);
    if(pdmac == 1)  sscanf(str, 50, "|%s", p->d_mac);
    if(pdata == 1)  sscanf(str, 50, "|%s", p->data);
}

int main(int argc, char **argv)
{
    int len, sfd, cfd, n;
    struct sockaddr_in saddr, caddr;
    char buffer[1024], content[1024];
    char *c_ip, *c_mac, *sent_pac;
    pid_t child;
    arp_packet *p = malloc(sizeof(arp_packet));

    //create a socket and get the socket fd
    sfd = socket(AF_INET, SOCK_STREAM, 0); //socket(int domain,int type,int
protocol)
    if(sfd<0)
        perror("Cannot create socket");

    //create server sockaddr and bind
    bzero(&saddr, sizeof(saddr));
    saddr.sin_family = AF_INET;
    saddr.sin_addr.s_addr = INADDR_ANY;
```

```
    saddr.sin_port = htons(PORT);

    connect(sfd, (struct sockaddr *)&saddr, sizeof(saddr));
    printf("Enter the IP Address: ");
    scanf("%s", c_ip);
    printf("\nEnter the MAC Address: ");
    scanf(" %s", c_mac);

    recv(sfd, buffer, sizeof(buffer), 0);
    printf("\nARP Request Received: ");
    stringtopac(p, buffer, 0, 0);
    print_arp_packet(p, 0, 0);

    if(strcmp(p->d_ip, c_ip) == 0)
    {
        printf("\nIP Address Match");
        strcpy(p->d_mac, c_mac);
        pactostring(p, buffer, 1, 0);
        strcpy(sent_pac, buffer);
        send(sfd, buffer, sizeof(buffer), 0);
        printf("\nARP Packet Sent:");
        print_arp_packet(p, 1, 0);
        while(strcmp(buffer, sent_pac) == 0) read(sfd, buffer,
sizeof(buffer));
        printf("\nRecieved Packet is: ");
        pactostring(p, buffer, 1, 1);
        print_arp_packet(p, 1, 1);
    }

    else printf("\nIP doesn't match!");
    close(sfd);
    return 0;
}
```