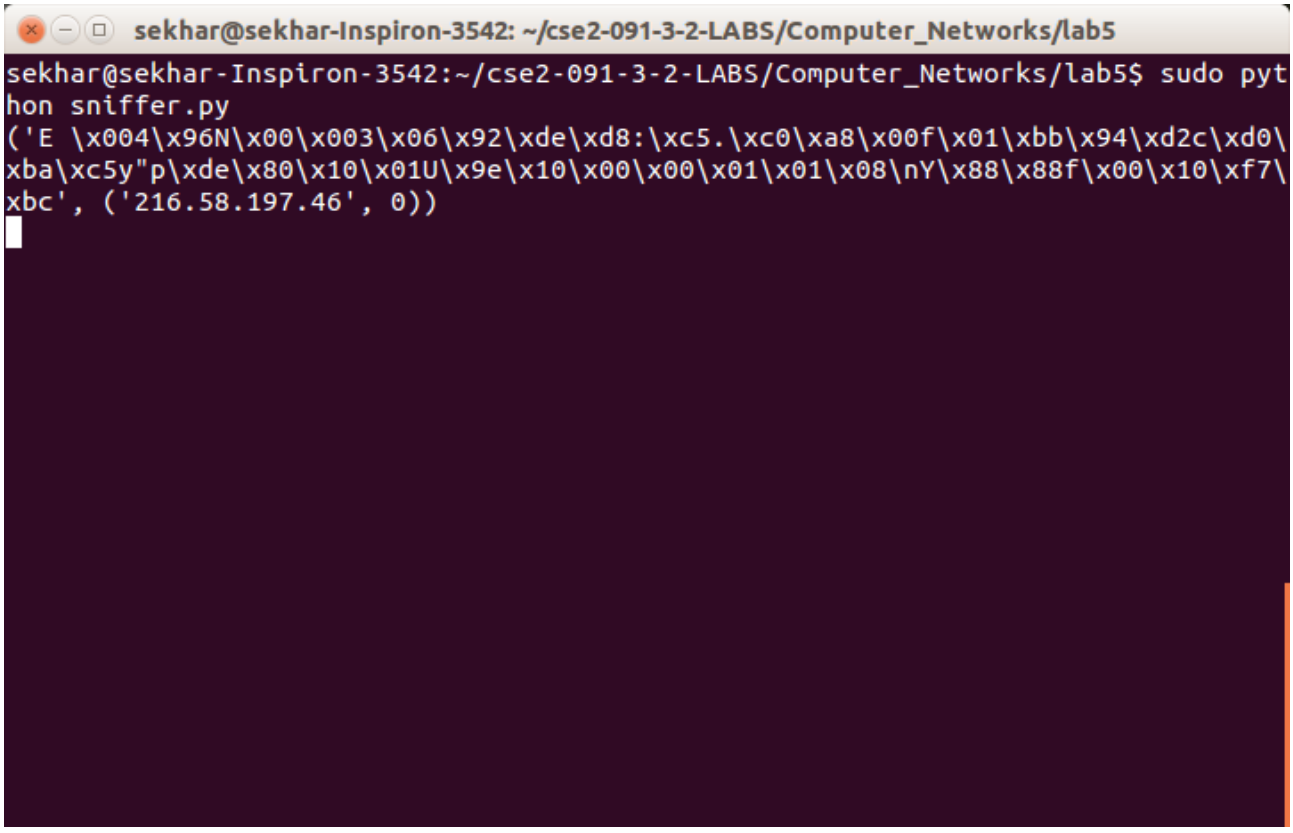
PROGRAM:-

```
#Packet sniffer in python
#For Linux
import socket
#create an INET, raw socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
# receive a packet
while True:
        print s.recvfrom(65565)
```

OUTPUT:-



PROGRAM:-

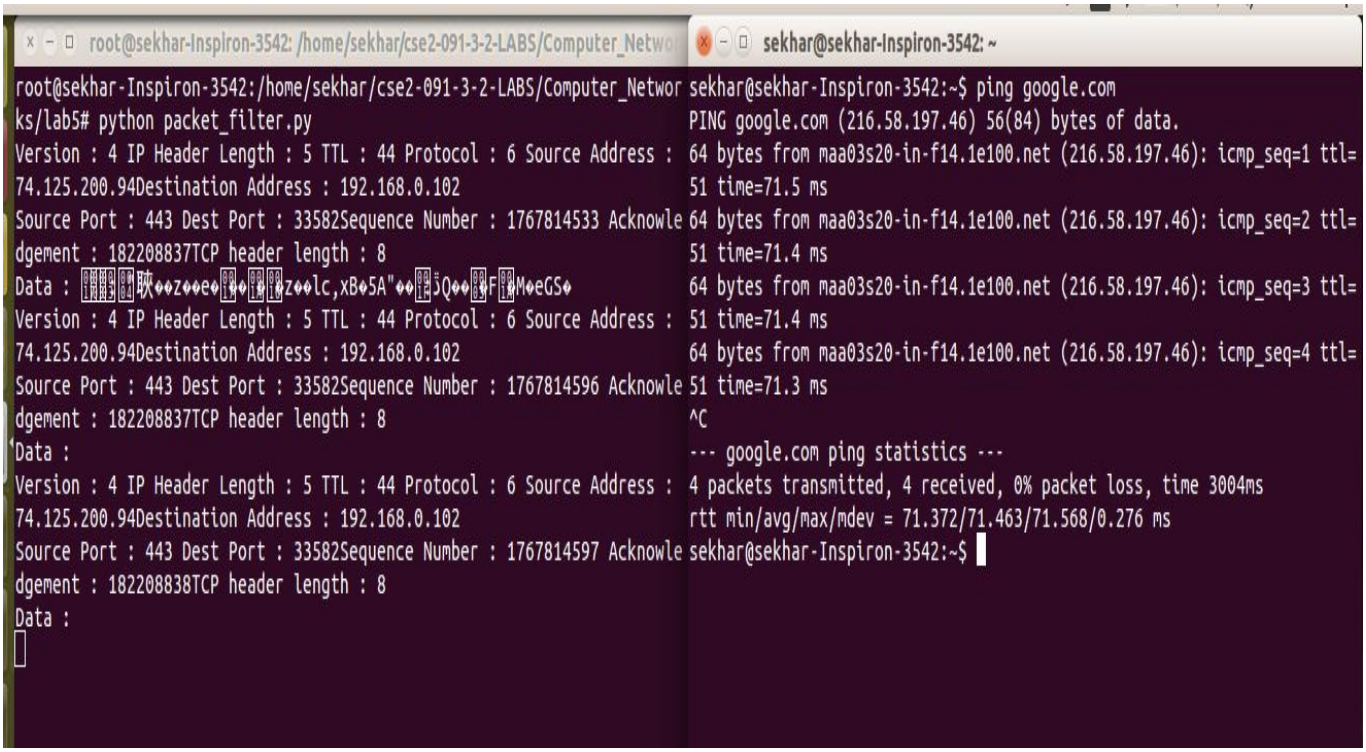
```
#Packet sniffer in python for Linux
#Sniffs only incoming TCP packet
import socket, sys
from struct import *
#create an INET, STREAMing socket
try:
	s = socket.socket(socket.AF_INET, socket.SOCK_RAW,
socket.IPPROTO_TCP)

except socket.error , msg:
	print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message '
+msg[1]
	sys.exit()
# receive a packet
while True:
	packet = s.recvfrom(65565)
#packet string from tuple
#	print "hi"
	packet = packet[0]
#take first 20 characters for the ip header
	ip_header = packet[0:20]
#now unpack them :)
	iph = unpack('!BBHHHBBH4s4s' , ip_header)
	version_ihl = iph[0]
	version = version_ihl >> 4
	ihl = version_ihl & 0xF
	iph_length = ihl * 4
	ttl = iph[5]
	protocol = iph[6]
	s_addr = socket.inet_ntoa(iph[8]);
	d_addr = socket.inet_ntoa(iph[9]);
	print 'Version : ' + str(version) + ' IP Header Length : ' + str(ihl) + ' TTL : '
+str(ttl) + ' Protocol : ' + str(protocol) + ' Source Address : ' + str(s_addr) +
'Destination Address : ' + str(d_addr)
	tcp_header = packet[iph_length:iph_length+20]
#now unpack them :)
	tcph = unpack('!HHLLBBHHH' , tcp_header)

	source_port = tcph[0]
	dest_port = tcph[1]
	sequence = tcph[2]
	acknowledgement = tcph[3]
```

```
        doff_reserved = tcph[4]
        tcph_length = doff_reserved >> 4
        print 'Source Port : ' + str(source_port) + ' Dest Port : ' + str(dest_port) +
'Sequence Number : ' + str(sequence) + ' Acknowledgement : ' +
str(acknowledgement) + 'TCP header length : ' + str(tcph_length)
        h_size = iph_length + tcph_length * 4
        data_size = len(packet) - h_size
#get data from the packet
        data = packet[h_size:]
        print 'Data : ' + data
```

OUTPUT:-



PROGRAM:-

SERVER

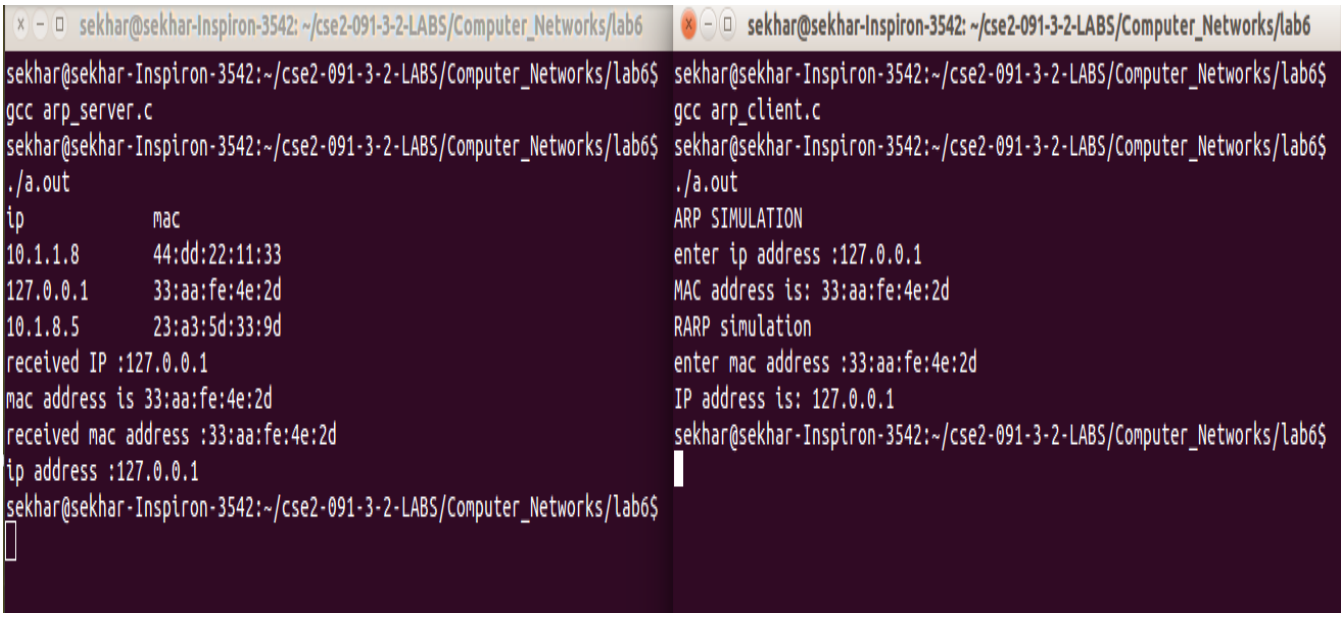
```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "sys/types.h"
#include "sys/socket.h"
#include "arpa/inet.h"
#include "netinet/in.h"
#define SA struct sockaddr
struct IPmac {
char ip[100];
char mac[100];
};
int main() {
int sockfd,len,i;
struct sockaddr_in servaddr;
char buff[30],temp[30],ip[30],mac[30];
int flag=0;
struct IPmac in[3]={
{"10.1.1.8","44:dd:22:11:33"},
{"127.0.0.1","33:aa:fe:4e:2d"},
{"10.1.8.5","23:a3:5d:33:9d"}
};
//printing table
printf("ip\t\tmac\n");
for(i=0;i<3;i++)
{
printf("%s\t%s\n",in[i].ip,in[i].mac);
}
//create socket
sockfd = socket(AF_INET,SOCK_DGRAM,0);
//fill structure
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(9999);
servaddr.sin_addr.s_addr = INADDR_ANY;
//bind
bind(sockfd,(SA*)&servaddr,sizeof(servaddr));
//get ip from client
len=sizeof(servaddr);
recvfrom(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,&len);
for(i=0;i<strlen(ip)-1;i++) {
```

```
temp[i]=ip[i];
}
temp[i]='\0';
printf("received IP :%s\n",temp);
//searching in table for equivalent mac
for(i=0;i<3;i++) {
if(strcmp(temp,in[i].ip)==0) {
strcpy(mac,in[i].mac);
break;
}
}
printf("mac address is %s\n",mac);
sendto(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,len);
//rarp simulation
//recv mac address
bzero(mac,sizeof(mac));
recvfrom(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,&len);
printf("received mac address :%s",mac);
//store in temp
bzero(temp,sizeof(temp));
for(i=0;i<strlen(mac)-1;i++) {
temp[i]=mac[i];
}
temp[i]='\0';
bzero(ip,sizeof(ip));
//check in table
for(i=0;i<3;i++) {
if(strcmp(temp,in[i].mac)==0) {
strcpy(ip,in[i].ip);
break;
}
}
printf("ip address :%s\n",ip);
sendto(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,len);
return 0;
}
```

CLIENT

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "sys/types.h"
#include "sys/socket.h"
#include "arpa/inet.h"
#include "netinet/in.h"
#define SA struct sockaddr
int main()
{
int sockfd,len;
char ip[30],mac[30];
struct sockaddr_in servaddr;
//creating socket
sockfd = socket(AF_INET,SOCK_DGRAM,0);
//fill structure
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(9999);
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
//send ip address
printf("ARP SIMULATION\n");
printf("enter ip address :");
fgets(ip,sizeof(ip),stdin);
sendto(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,sizeof(servaddr));
len=sizeof(servaddr);
recvfrom(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,&len);
printf("MAC address is: %s\n",mac);
printf("RARP simulation\n");
printf("enter mac address :");
bzero(mac,sizeof(mac));
fgets(mac,sizeof(mac),stdin);
sendto(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,len);
recvfrom(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,&len);
printf("IP address is: %s\n",ip);
return 0;
}
```

OUTPUT:-


sekhar@sekhar-Inspiron-3542: ~/cse2-091-3-2-LABS/Computer_Networks/lab6
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$ gcc arp_server.c
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$ ./a.out
ip mac
10.1.1.8 44:dd:22:11:33
127.0.0.1 33:aa:fe:4e:2d
10.1.8.5 23:a3:5d:33:9d
received IP :127.0.0.1
mac address is 33:aa:fe:4e:2d
received mac address :33:aa:fe:4e:2d
ip address :127.0.0.1
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$
sekhar@sekhar-Inspiron-3542: ~/cse2-091-3-2-LABS/Computer_Networks/lab6
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$ gcc arp_client.c
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$ ./a.out
ARP SIMULATION
enter ip address :127.0.0.1
MAC address is: 33:aa:fe:4e:2d
RARP simulation
enter mac address :33:aa:fe:4e:2d
IP address is: 127.0.0.1
sekhar@sekhar-Inspiron-3542:~/cse2-091-3-2-LABS/Computer_Networks/lab6$