Q.1 Run the program given in UDP_Socket document (Example 6.1,6.2,6.3 and 6.4).

Output:

// client

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

int main()

{
int sid;
char c;
struct sockaddr_in server_address;
int server_addlen;
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 7890;
server_addlen = sizeof(server_address);

sid = socket(AF_INET, SOCK_DGRAM, 0);

sendto(sid, "A", 1, 0, (struct sockaddr *)&server_address, server_addlen);
recvfrom(sid, &c, 1, 0, (struct sockaddr *)&server_address, &server_addlen);
printf("character from server is %c \n", c);
return (0);
}
```

// server

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

int main()

{
int sid;
char c;
struct sockaddr_in server_address, client_address;
int server_addlen, cli_addlen;
server_address.sin_family = AF_INET;
```

```
 server_address.sin_addr.s_addr = inet_addr("127.0.0.1");

 server_address.sin_port = 7890;

 server_addlen = sizeof(server_address);
 cli_addlen = sizeof(client_address);

 sid = socket(AF_INET, SOCK_DGRAM, 0);
 bind(sid, (struct sockaddr *)&server_address, server_addlen);

 while (1)
 {
 printf("Ready to recv datagram \n");
 recvfrom(sid, &c, 1, 0, (struct sockaddr *)&client_address, &cli_addlen);

 sendto(sid, &c, 1, 0, (struct sockaddr *)&client_address, cli_addlen);
 }

 return (0);
 }
```

Q.2 Execute a client/server program using UDP service for adding a two integer numbers requested by the client and evaluated at server and get back result at the client.

Output:

// client

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

int main()

{
int sid;
int arr[2];
scanf("%d%d",&arr[0],&arr[1]);
struct sockaddr_in server_address;
int server_addlen;
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 7890;
server_addlen = sizeof(server_address);

sid = socket(AF_INET, SOCK_DGRAM, 0);
int sum;
sendto(sid, arr, sizeof(arr), 0, (struct sockaddr *)&server_address,
server_addlen);
recvfrom(sid, &sum, sizeof(sum), 0, (struct sockaddr *)&server_address,
&server_addlen);


printf("Sum from server is %d \n", sum);
return (0);
}
```

//server

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

int main()

{
int sid;
```

```
int arr[2];
int sum;
struct sockaddr_in server_address, client_address;
int server_addlen, cli_addlen;
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 7890;

server_addlen = sizeof(server_address);
cli_addlen = sizeof(client_address);

sid = socket(AF_INET, SOCK_DGRAM, 0);
bind(sid, (struct sockaddr *)&server_address, server_addlen);

while (1)
{
printf("Ready to recv datagram \n");
recvfrom(sid, &arr, sizeof(arr), 0, (struct sockaddr *)&client_address,
&cli_addlen);

sum=arr[0]+arr[1];
sendto(sid, &sum, sizeof(sum), 0, (struct sockaddr *)&client_address,
cli_addlen);
}

return (0);
}
```

Q.3 Execute a client/server program for simple calculator having following operations addition, subtraction, multiplication, division, and detecting prime numbers. Input entered at the client side and evaluated at server machine and get back result at the client machine. Try this question for both TCP and UDP socket programming.

Output:

Clients

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

int main()
{
int sid;
int arr[3];
int res[5];
struct sockaddr_in server_address;
int server_addlen;
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 7890;

server_addlen = sizeof(server_address);

sid = socket(AF_INET, SOCK_DGRAM, 0);
printf("If u want caluclator press 1\n");
printf("If u want to check a number is prime or not press 0\n");
int typ;
scanf("%d", &typ);
if (typ == 1)
{
arr[2] = 1;
printf("Give the values of two numbers to perform caluclations\n");
scanf("%d%d", &arr[0], &arr[1]);
sendto(sid, arr, sizeof(arr), 0, (struct sockaddr *)&server_address,
server_addlen);
recvfrom(sid, res, sizeof(res), 0, (struct sockaddr *)&server_address,
&server_addlen);
int x;
printf("press 0 for sum, 1 for diff, 2 for multiplication , 3 for division\
n");
scanf("%d", &x);
switch (x)
```

```
{
case 0:
printf("Sum from server is %d \n", res[0]);
break;
case 1:
printf("difference from server is %d \n", res[1]);
break;
case 2:
printf("mult from server is %d \n", res[2]);
break;
case 3:

printf("div from server is %d \n", res[3]);
break;

default:
printf("Give a valid choice rerun \n");
}
}
if (typ == 0)

{
int num;
int check;
arr[2] = 0;
printf("Give the value a number to check whehter it is prime or not\n");
scanf("%d", &arr[0]);
sendto(sid, &arr, sizeof(arr), 0, (struct sockaddr *)&server_address,
server_addlen);
recvfrom(sid, &res, sizeof(res), 0, (struct sockaddr *)&server_address,
&server_addlen);
if (res[4] == 1)
{
printf("Yes it's prime!\n");
}
else
printf("Not prime!\n");
}

return (0);
}
```

//server

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

void cal(int arr[],int res[])
{
res[0]=arr[0]+arr[1];
res[1]=arr[0]-arr[1];
res[2]=arr[0]*arr[1];
res[3]=arr[0]/arr[1];
}
void primeCheck(int arr[],int res[])
{
int num= arr[0];
printf("came to check prime \n");
for(int i=2;i<num;i++)

{
if(num%i==0)
{
res[4]=0;
return;
}
}
res[4]=1;
}
int main()

{
int sid;
int arr[3];
int res[5];
struct sockaddr_in server_address, client_address;
int server_addlen, cli_addlen;
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 7890;

server_addlen = sizeof(server_address);
cli_addlen = sizeof(client_address);

sid = socket(AF_INET, SOCK_DGRAM, 0);
bind(sid, (struct sockaddr *)&server_address, server_addlen);

while (1)
{
printf("Ready to recv datagram \n");
recvfrom(sid, &arr, sizeof(arr), 0, (struct sockaddr *)&client_address,
```

```
&cli_addlen);

if(arr[2]==1)
cal(arr,res);
if(arr[2]==0)
primeCheck(arr,res);

sendto(sid, &res, sizeof(res), 0, (struct sockaddr *)&client_address,
cli_addlen);
}

return (0);
}
```