NAME – ASHUTOSH ARDU

REG NO. – 20BRS1262

DATE – 27-4-2021

CSE1004 UDP AND TCP SOCKET PROGRAMMING

USING UDP PROTOCOL

FINDING OUT THE SUM OF 'N' WHOLE NUMBERS

ALGORITHM

- FIRST A SERVER IS CREATED WHICH CAN CALCULATE THE SUM OF 'N' NATURAL NUMBERS AND THEN A SOCKET IS CREATED, LATER THE SERVER IS BINDED TO THAT SOCKET AND IS NOW READY FOR FURTHER OPERATIONS.

- A CLIENT JOINS THE SOCKET AND SENDS A QUERY WHICH INCLUDES A NUMBER 'N' FOR WHICH IT WANTS THE SUM.

- AFTER THE SERVER RECEIVES THE QUERY, CALCULATES THE SUM OF 'N' USING A FUNCTION AND THE FORMULA -N*(N-1)/2- AND SENDS THE SUM AS QUERY BACK TO THE CLIENT.

- THE CLIENT RECEIVES THE QUERY AND HENCE THE SOCKET IS CLOSED.

$\downarrow$

THE CODE

THE CLIENT SIDE

```c
// UDP client and UDP server
// Client Side
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>

int main(){
    int port,sock,binding,numrecv,numsend;
    struct sockaddr_in server,client;
    char buffersent[100],bufferrecv[100];
    socklen_t serverlen;
    serverlen=sizeof(server);
    sock=socket(AF_INET,SOCK_DGRAM,0);
    printf("Enter the Number\n");
    scanf("%d",&numsend);
    printf("Enter the port number\n");
    scanf("%d",&port);
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=htonl(INADDR_ANY);
    server.sin_port=htons(port);
    sendto(sock,&numsend,sizeof(numsend),0,(struct sockaddr*)&
server,sizeof(server));
    recvfrom(sock,&numrecv,sizeof(numrecv),0,(struct sockaddr*)
&server,&serverlen);
    printf("The Sum of the given number\n%d\n",numrecv);
}
```

THE SERVER SIDE

```c
// UDP client and UDP server
// Server Side
#include<stdio.h>
#include<sys/socket.h>
```

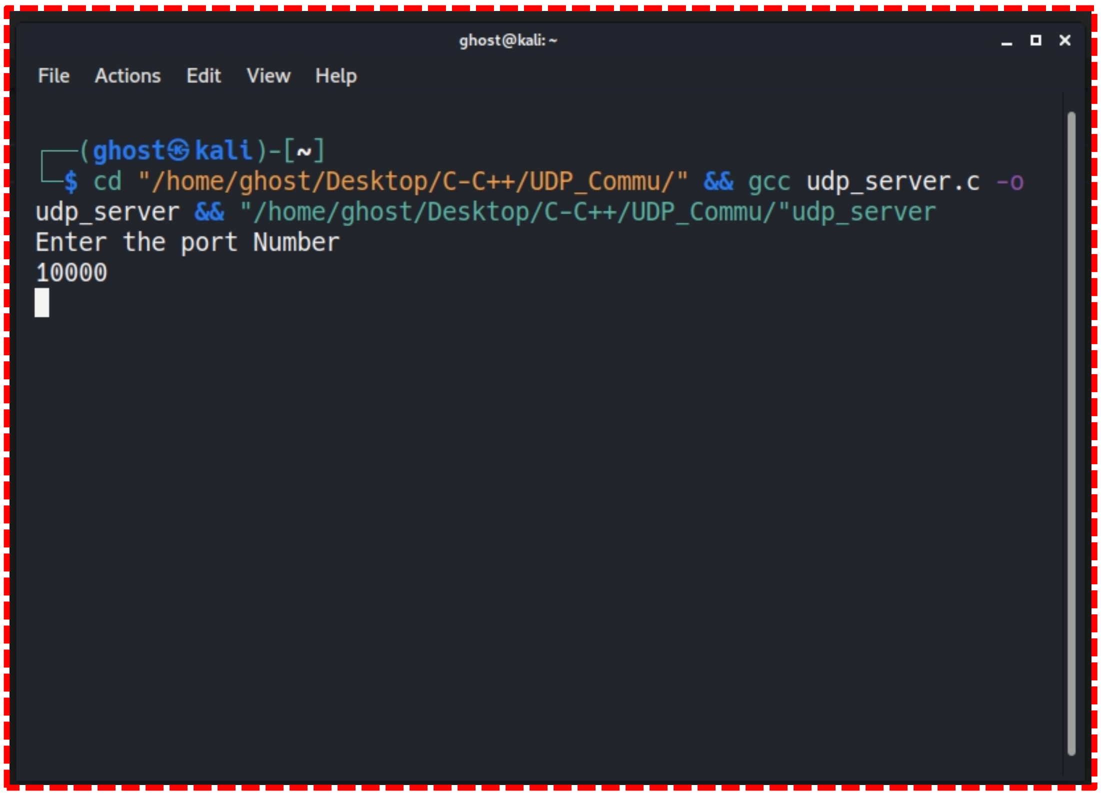
```
#include<netinet/in.h>
#include<string.h>

int sum(int num){
    int out=0;
    if(num<0)
        return 0;
    else
        out=num*(num-1)/2;
    return out;
}

int main(){
    int port,sock,binding,numrecv,numsend;
    struct sockaddr_in server,client;
    char buffersent[100],bufferrecv[100];
    socklen_t clientlen;
    sock=socket(AF_INET,SOCK_DGRAM,0);
    printf("Enter the port Number\n");
    scanf("%d",&port);
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=htonl(INADDR_ANY);
    server.sin_port=htons(port);
    binding=bind(sock,(struct sockaddr*)&server,sizeof(se
rver));
    clientlen=sizeof(client);
    recvfrom(sock,&numrecv,sizeof(numrecv),0,(struct sock
addr*)&client,&clientlen);
    printf("The number received\n%d\n",numrecv);
    numsend=sum(numrecv);
    sendto(sock,&numsend,sizeof(numsend),0,(struct sockad
dr*)&client,sizeof(client));
}
```

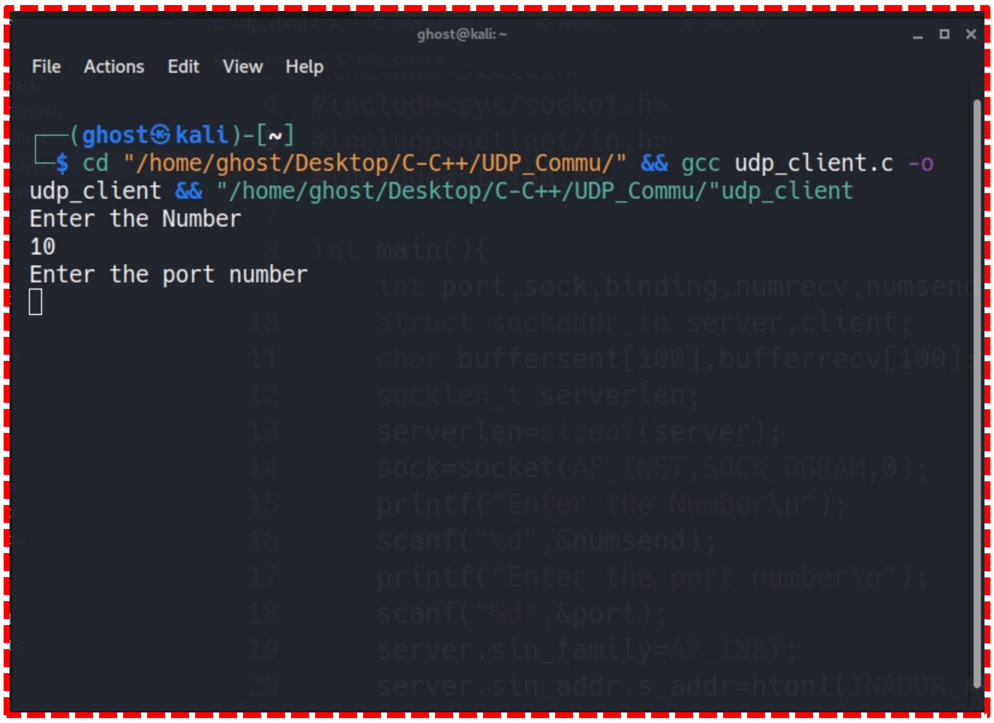
OUTPUTS

FIRSTLY, A SOCKET IS CREATED AND THE SERVER IS BINDED TO IT USING A PORT NUMBER.



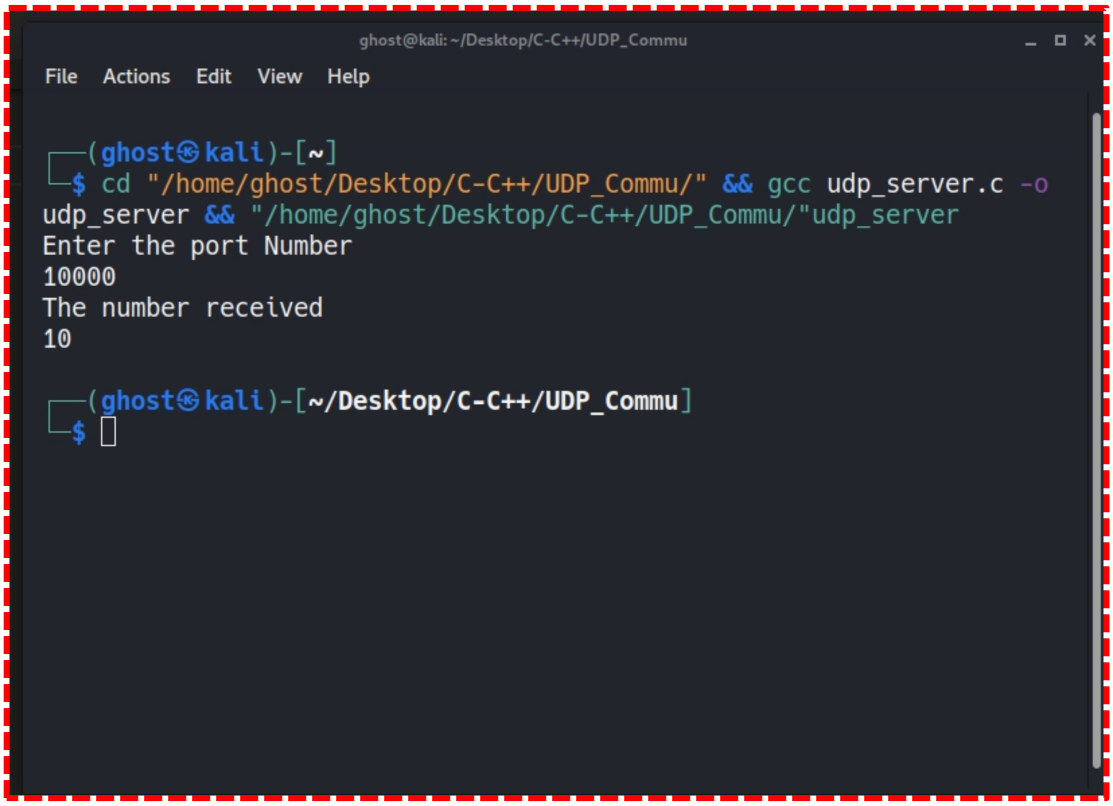
```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/UDP_Commu/" && gcc udp_server.c -o
udp_server && "/home/ghost/Desktop/C-C++/UDP_Commu/"udp_server
Enter the port Number
10000
```

NOW THE CLIENT STORES THE VALUE OF ‘N’
AND CONNECTS TO THE SERVER’S PORT NUMBER



```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/UDP_Commu/" && gcc udp_client.c -o
udp_client && "/home/ghost/Desktop/C-C++/UDP_Commu/"udp_client
Enter the Number
10
Enter the port number
```

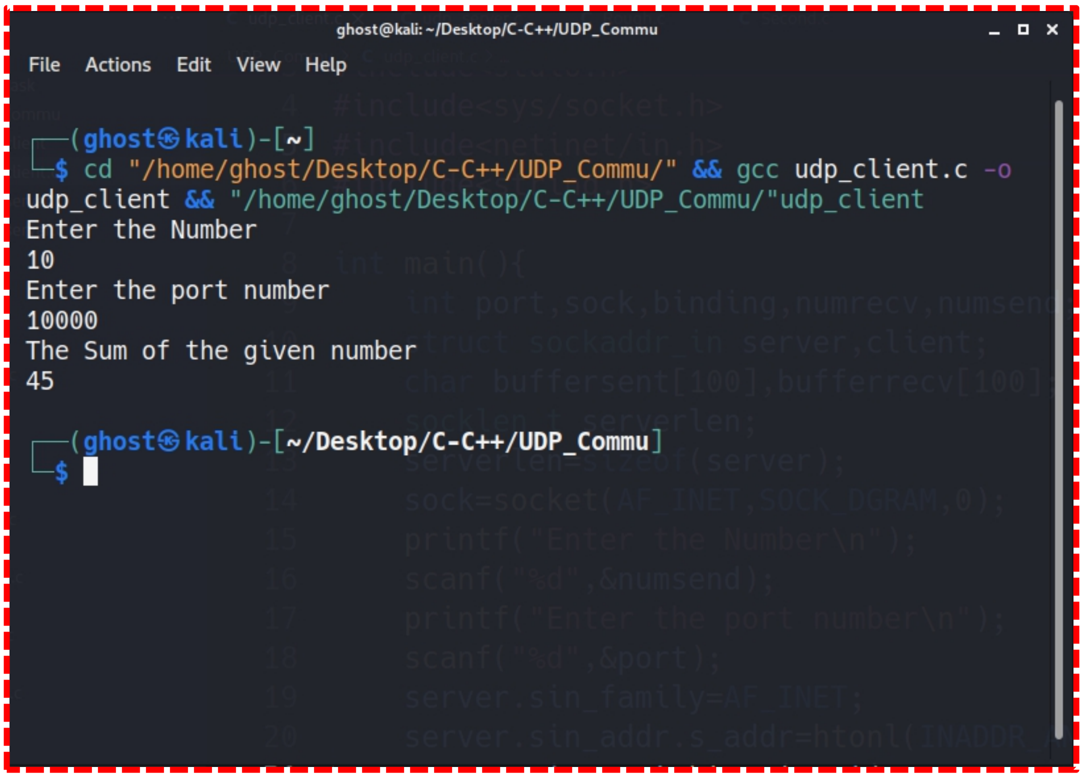
SERVER RECEIVES THE NUMBER AND COMPUTES THE SUM OF 'N' NATURAL NUMBERS
AND SENDS THE SUM BACK TO THE CLIENT


ghost@kali: ~/Desktop/C-C++/UDP_Commu
File Actions Edit View Help

```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/UDP_Commu/" && gcc udp_server.c -o
udp_server && "/home/ghost/Desktop/C-C++/UDP_Commu/"udp_server
Enter the port Number
10000
The number received
10

┌──(ghost㉿kali)-[~/Desktop/C-C++/UDP_Commu]
└─$ 
```

THE CLIENT RECEIVES THE SUM AND THE THUS THE SOCKET IS CLOSED


ghost@kali: ~/Desktop/C-C++/UDP_Commu
File Actions Edit View Help

```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/UDP_Commu/" && gcc udp_client.c -o
udp_client && "/home/ghost/Desktop/C-C++/UDP_Commu/"udp_client
Enter the Number
10
Enter the port number
10000
The Sum of the given number
45

┌──(ghost㉿kali)-[~/Desktop/C-C++/UDP_Commu]
└─$ 
```

TCP PROTOCOL

CHECKING WHETHER THE NUMBER IS A PRIME OR NOT

ALGORITHM

- FIRST A SERVER IS CREATED WHICH CAN CHECK WHETHER A NUMBER IS PRIME OR NOT AND THEN A SOCKET IS CREATED, LATER THE SERVER IS BINDED TO THAT SOCKET AND IS NOW READY FOR FURTHER OPERATIONS.

- A CLIENT JOINS THE SOCKET AND SENDS A QUERY WHICH INCLUDES A NUMBER 'N' FOR WHICH IT WANTS THE VERIFICATION WHETHER IT IS PRIME OR NOT.

- THE SERVER RECEIVES THE NUMBER 'N' AND CHECKS WHETHER IT IS PRIME OR NOT AND SENDS THE VERIFICATION BACK TO THE CLIENT.

- THE CLIENT RECEIVES THE VERIFICATION AND THE SOCKET IS CLOSED.

CODE

THE CLIENT SIDE

```
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
// Client Side
int main()
{
    int soc,port,numsend;
    char mess[100],mess2[100];
    struct sockaddr_in server,client;
    printf("Enter the Number\n");
    scanf("%d",&numsend);
    printf("Enter the port no.\n");
    scanf("%d",&port);
    soc=socket(AF_INET,SOCK_STREAM,0);
    if(soc<0)
```

```
        printf("Socket not created\n");
    else
        printf("Socket Created\n");
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=htonl(INADDR_ANY);
    server.sin_port=htons(port);

    if(connect(soc,(struct sockaddr*)&server,sizeof(server))<0)
        printf("Can't connect\n");
    else
        printf("Connected\n");

    send(soc,&numsend,sizeof(numsend),0);
    recv(soc,mess2,sizeof(mess2),0);
    printf("%s",mess2);
    return 0;
}
```

THE SERVER SIDE

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

int prime(int num){
    int out=0;
    if(num==0 || num==1 || num<0)
        return 0;
    else if(num==2)
        return 1;
    else{
        for(int i=2;i<num;++i){
            if(num%i==0)
```

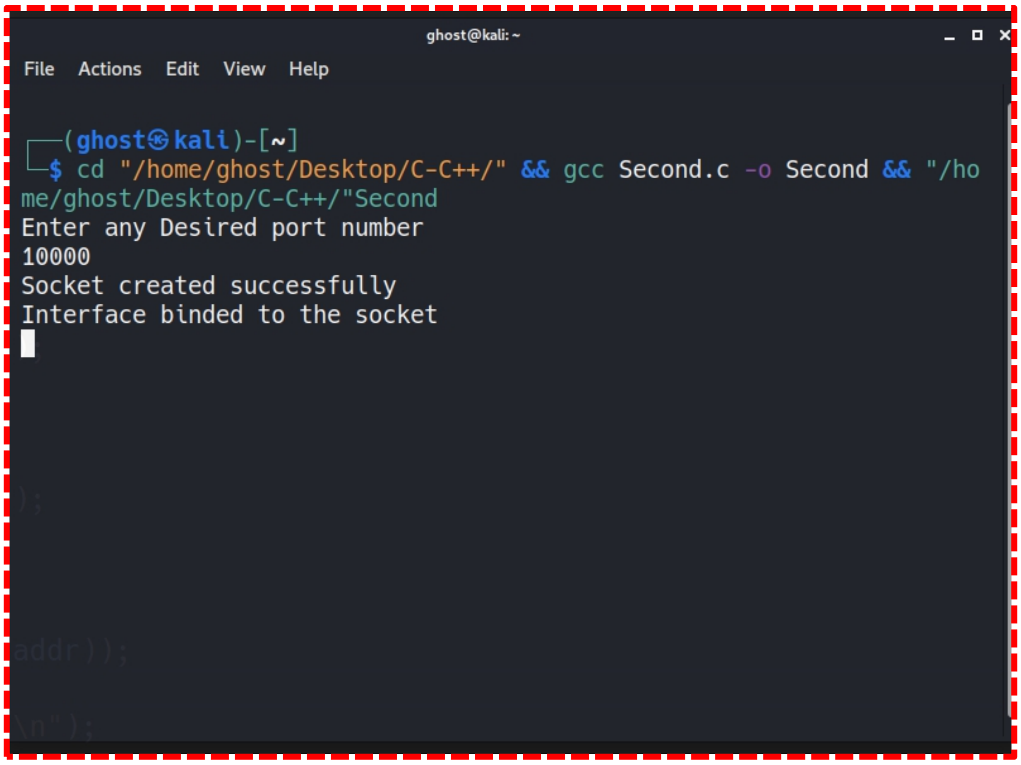
```
                ++out;
            }
            if(!out)
                return 1;
            else
                return 0;
        }
    }

    int main()
    {
        int s,b,sport,key,numrecv;
        char mess[100],mess2[100];
        struct sockaddr_in saddr,caddr;
        printf("Enter any Desired port number\n");
        scanf("%d",&sport);
        int clen=sizeof(caddr);
        s=socket(AF_INET,SOCK_STREAM,0);
        if(s<0)
        {
            printf("Error while socket creation\n");
        }

        else
            printf("Socket created successfully\n");
        saddr.sin_family=AF_INET;
        saddr.sin_addr.s_addr=htonl(INADDR_ANY);
        saddr.sin_port=htons(sport);
        b=bind(s,(struct sockaddr*)&saddr,sizeof(saddr));
        if(b==0)
            printf("Interface binded to the socket\n");
        else
            printf("Interface not binded to the socket\n");
        listen(s,5);
        key=accept(s,(struct sockaddr*)&caddr,&clen);
        if(key<0)
            printf("Error\n");
        else
        printf("Well Connected\n");
    recv(key,&numrecv,sizeof(numrecv),0);
    printf("The received number\n%d\n",numrecv);
    int k=prime(numrecv);
    if(k==1)
        strcpy(mess2,"YES IT IS A PRIME");
    else
        strcpy(mess2,"NO NOT A PRIME");
    send(key,mess2,sizeof(mess2),0);
}
```

OUTPUTS

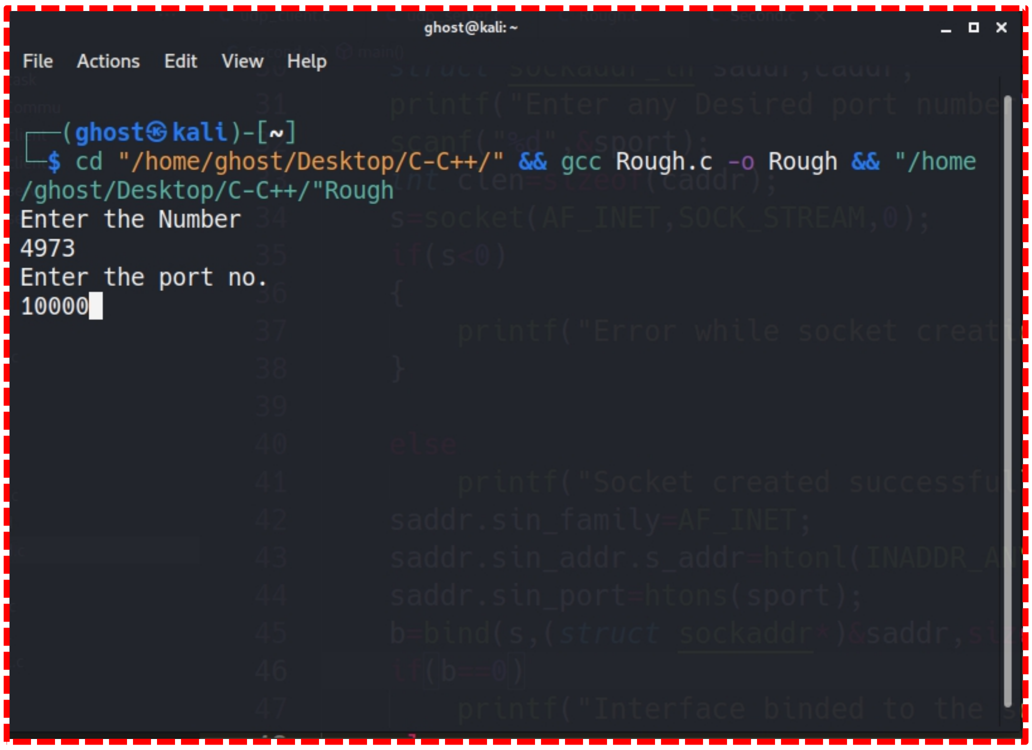
A SOCKET IS CREATED AND THE SERVER IS BINDED TO THAT SOCKET



```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/" && gcc Second.c -o Second && "/ho
me/ghost/Desktop/C-C++/"Second
Enter any Desired port number
10000
Socket created successfully
Interface binded to the socket
```

CLIENT CONNECTS TO THE SOCKET AND SENDS THE NUMBER 'N'

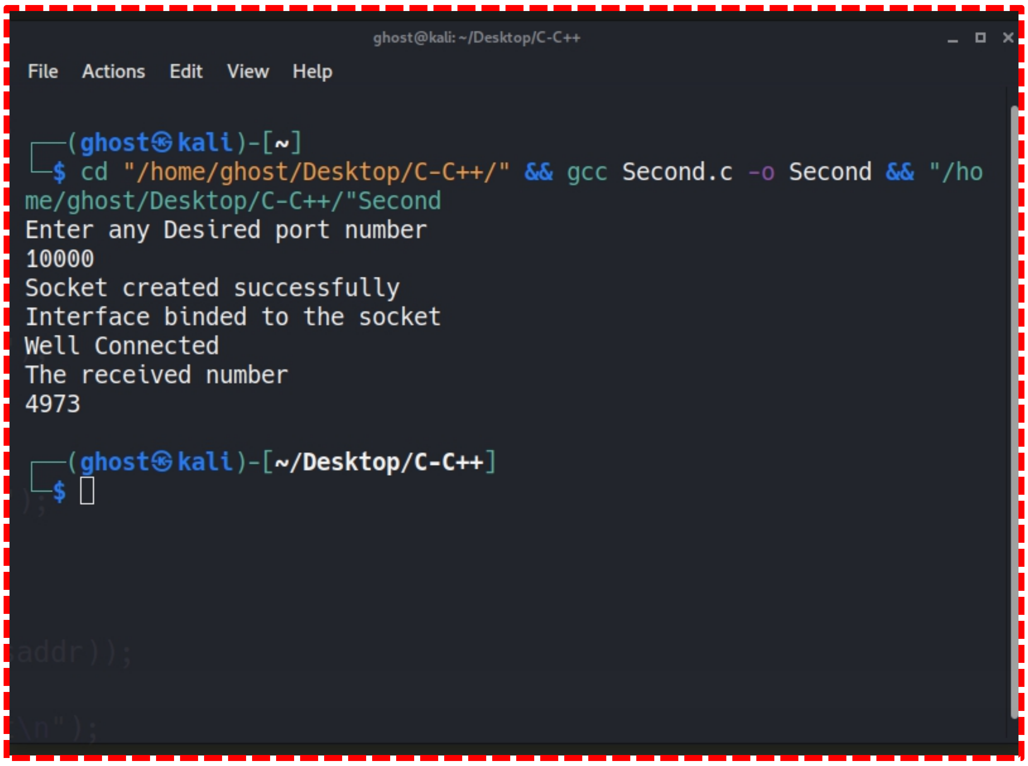
AS THE QUERY



```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/" && gcc Rough.c -o Rough && "/home
/ghost/Desktop/C-C++/"Rough
Enter the Number
4973
Enter the port no.
10000
```

THE SERVER RECEIVES THE VALUE AND CHECKS WHETHER THE GIVEN NUMBER

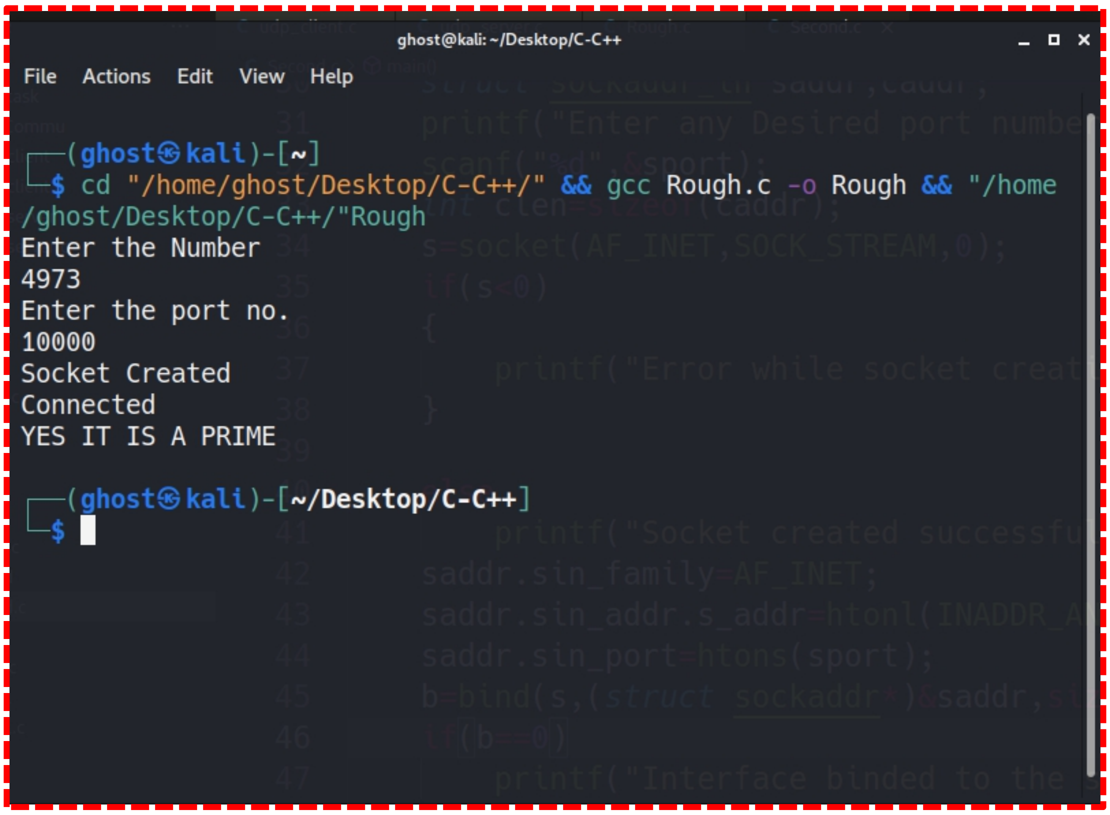
IS PRIME OR NOT AND SEND THE VERIFICATION BACK TO THE CLIENT



```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/" && gcc Second.c -o Second && "/ho
me/ghost/Desktop/C-C++/"Second
Enter any Desired port number
10000
Socket created successfully
Interface binded to the socket
Well Connected
The received number
4973

┌──(ghost㉿kali)-[~/Desktop/C-C++]
└─$ 
```

THE CLIENT RECEIVES THE VERIFICATION AND THUS THE SOCKET IS CLOSED



```
┌──(ghost㉿kali)-[~]
└─$ cd "/home/ghost/Desktop/C-C++/" && gcc Rough.c -o Rough && "/home/ghost/Desktop/C-C++/"Rough
Enter the Number
4973
Enter the port no.
10000
Socket Created
Connected
YES IT IS A PRIME

┌──(ghost㉿kali)-[~/Desktop/C-C++]
└─$ 
```