MAYANK SINHA
RA1911003010386
EXP 8

SERVER:

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

#define IP_PROTOCOL 0
#define PORT_NO 15055
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"


void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}


char Cipher(char ch)
{
    return ch ^ cipherKey;
}


int sendFile(FILE* fp, char* buf, int s)
{
    int i, len;
    if (fp == NULL) {
        strcpy(buf, nofile);
        len = strlen(nofile);
        buf[len] = EOF;
```

```
        for (i = 0; i <= len; i++)
            buf[i] = Cipher(buf[i]);
        return 1;
    }

    char ch, ch2;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
        ch2 = Cipher(ch);
        buf[i] = ch2;
        if (ch == EOF)
            return 1;
    }
    return 0;
}


int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = INADDR_ANY;
    char net_buf[NET_BUF_SIZE];
    FILE* fp;


    sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);

    if (sockfd < 0)
        printf("\nfile descriptor not received!!\n");
    else
        printf("\nfile descriptor %d received\n", sockfd);


    if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
        printf("\nSuccessfully binded!\n");
    else
        printf("\nBinding Failed!\n");

    while (1) {
        printf("\nWaiting for file name...\n");
```

```
        clearBuf(net_buf);

        nBytes = recvfrom(sockfd, net_buf,
                    NET_BUF_SIZE, sendrecvflag,
                    (struct sockaddr*)&addr_con, &addrlen);

        fp = fopen(net_buf, "r");
        printf("\nFile Name Received: %s\n", net_buf);
        if (fp == NULL)
            printf("\nFile open failed!\n");
        else
            printf("\nFile Successfully opened!\n");

        while (1) {


            if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
                sendto(sockfd, net_buf, NET_BUF_SIZE,
                      sendrecvflag,
                    (struct sockaddr*)&addr_con, addrlen);
                break;
            }


            sendto(sockfd, net_buf, NET_BUF_SIZE,
                  sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
            clearBuf(net_buf);
        }
        if (fp != NULL)
            fclose(fp);
    }
    return 0;
}
```

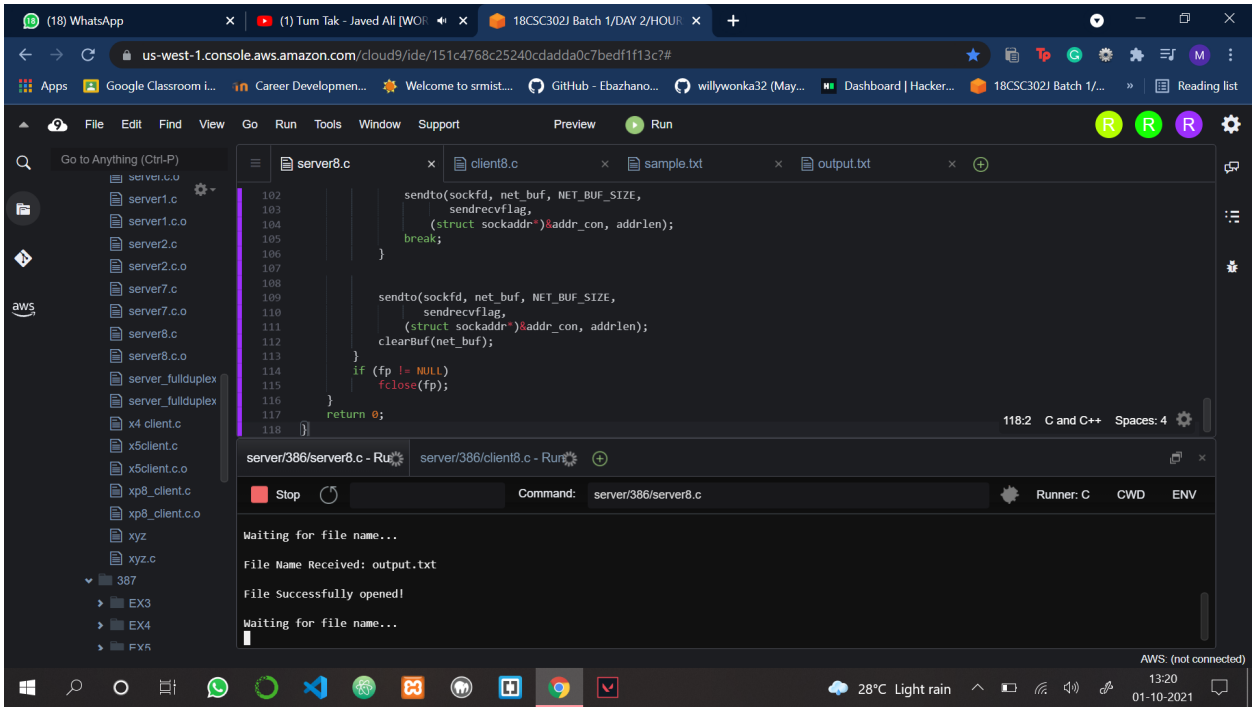
CLINET:

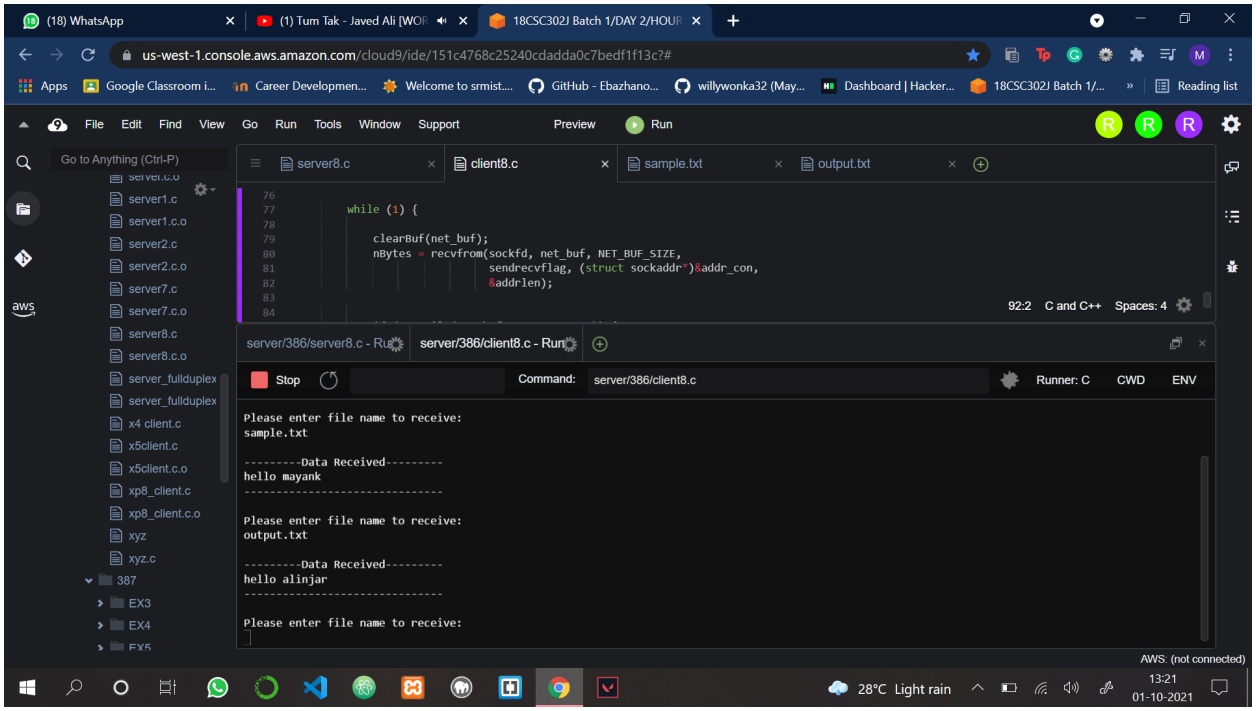
```
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1"
#define PORT_NO 15055
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0


void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}


char Cipher(char ch)
{
    return ch ^ cipherKey;
}


int recvFile(char* buf, int s)
{
    int i;
    char ch;
    for (i = 0; i < s; i++) {
        ch = buf[i];
        ch = Cipher(ch);
        if (ch == EOF)
            return 1;
        else
            printf("%c", ch);
    }
    return 0;
}


int main()
```

```
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
    char net_buf[NET_BUF_SIZE];
    FILE* fp;


    sockfd = socket(AF_INET, SOCK_DGRAM,
                    IP_PROTOCOL);

    if (sockfd < 0)
        printf("\nfile descriptor not received!!\n");
    else
        printf("\nfile descriptor %d received\n", sockfd);

    while (1) {
        printf("\nPlease enter file name to receive:\n");
        scanf("%s", net_buf);
        sendto(sockfd, net_buf, NET_BUF_SIZE,
               sendrecvflag, (struct sockaddr*)&addr_con,
               addrlen);

        printf("\n---------Data Received---------\n");

        while (1) {

            clearBuf(net_buf);
            nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
                              sendrecvflag, (struct sockaddr*)&addr_con,
                              &addrlen);


            if (recvFile(net_buf, NET_BUF_SIZE)) {
                break;
            }
        }
        printf("\n------------------------------\n");
    }
    return 0;
}
```

OUTPUT:


server8.c
client8.c
sample.txt
output.txt
sendto(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag,
(struct sockaddr*)&addr_con, addrlen);
break;
}
sendto(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag,
(struct sockaddr*)&addr_con, addrlen);
clearBuf(net_buf);
}
if (fp != NULL)
fclose(fp);
}
return 0;
}
118:2 C and C++ Spaces: 4
server/386/server8.c - Run
server/386/client8.c - Run
Stop
Command: server/386/server8.c
Runner: C CWD ENV
Waiting for file name...
File Name Received: output.txt
File Successfully opened!
Waiting for file name...
AWS: (not connected)
28°C Light rain
13:20
01-10-2021


server8.c
client8.c
sample.txt
output.txt
while (1) {
clearBuf(net_buf);
nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag, (struct sockaddr*)&addr_con,
&addrlen);
92:2 C and C++ Spaces: 4
server/386/server8.c - Run
server/386/client8.c - Run
Stop
Command: server/386/client8.c
Runner: C CWD ENV
Please enter file name to receive:
sample.txt
---------Data Received---------
hello mayank
Please enter file name to receive:
output.txt
---------Data Received---------
hello alinjar
Please enter file name to receive:
AWS: (not connected)
28°C Light rain
13:21
01-10-2021