NETWORKS LAB 2
ROLL NUMBER: 106119109
NAME: SALONI RAKHOLIYA

QUESTION 1:
Explanation: taking in a range of port values to check, and then trying to connect to all to see if open or not;
CODE:
SCANNER:

```c
// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

int main()
{
   int sock = 0, valread;
   struct sockaddr_in serv_addr;
   char *hello = "Hello from client";
   char buffer[1024] = {0};
   int start,end;
   printf("Enter starting port number for searching: ");
   scanf("%d",&start);
   printf("Enter ending port number for searching: ");
   scanf("%d",&end);
   printf("\n");
   for(int i=start;i<end;++i)
   {

   if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
   {
       printf("\n Socket creation error \n");
       return -1;
   }
```

```
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(i);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0)
    {
        //printf("\nPort %d closed\n",i);
    }
    else {
        printf("Port %d open\n",i);
    }
    }
    return 0;
}
```

SERVER 1:

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
```

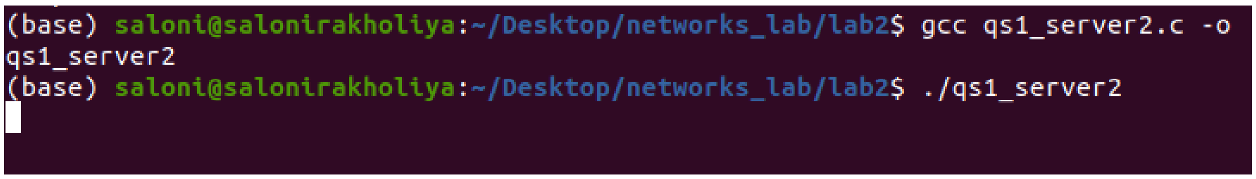
```c
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                  &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                    (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    return 0;
 }
```

SERVER 2:

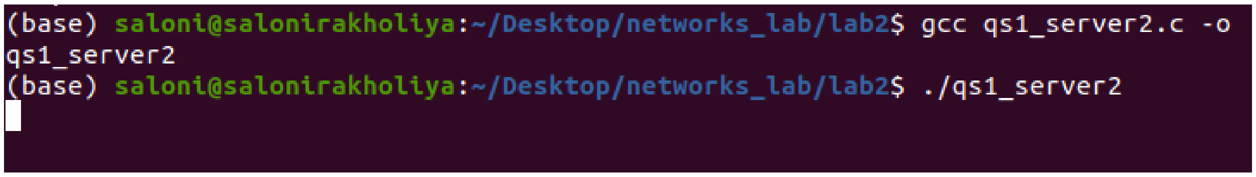
```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8000
int main(int argc, char const *argv[])
{
   int server_fd, new_socket, valread;
   struct sockaddr_in address;
   int opt = 1;
   int addrlen = sizeof(address);
   char buffer[1024] = {0};
   char *hello = "Hello from server";

   // Creating socket file descriptor
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
   {
       perror("socket failed");
       exit(EXIT_FAILURE);
   }

   // Forcefully attaching socket to the port 8080
   if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                  &opt, sizeof(opt)))
```

```c
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                    (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    return 0;
}
```

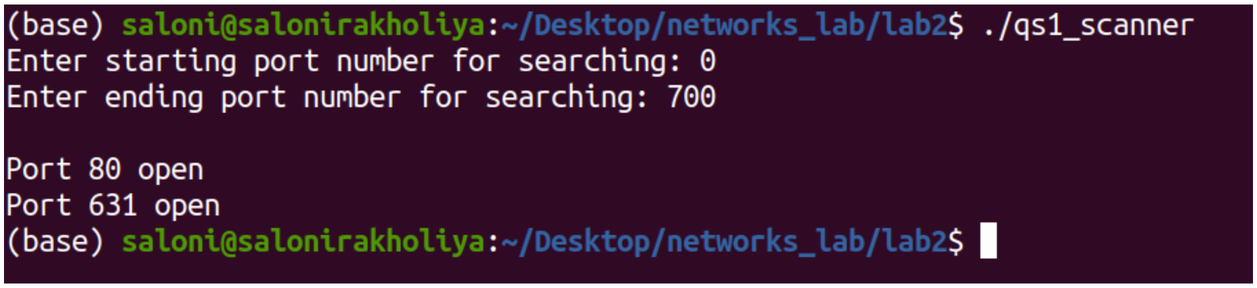
RESULTS:



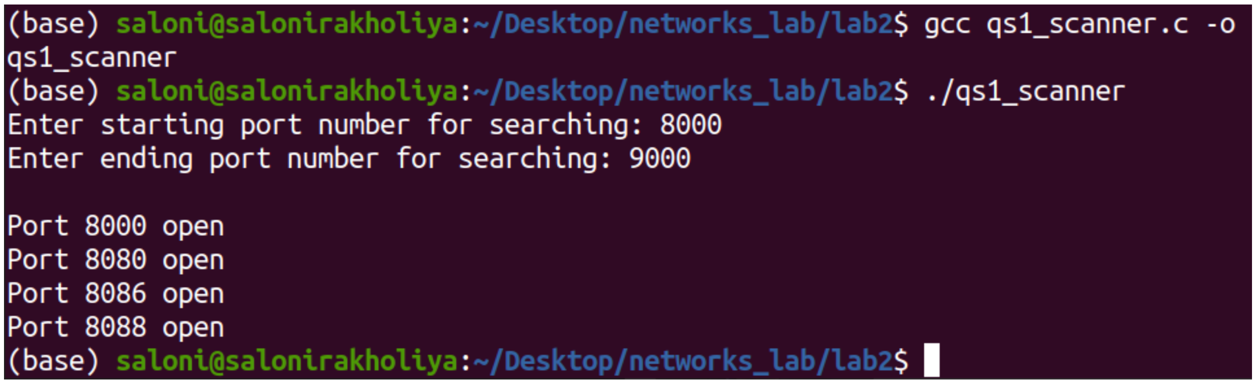
```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs1_server2.c -o
qs1_server2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs1_server2
```



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs1_server2.c -o
qs1_server2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs1_server2
```



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs1_scanner
Enter starting port number for searching: 0
Enter ending port number for searching: 700

Port 80 open
Port 631 open
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs1_scanner.c -o
qs1_scanner
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs1_scanner
Enter starting port number for searching: 8000
Enter ending port number for searching: 9000

Port 8000 open
Port 8080 open
Port 8086 open
Port 8088 open
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

QUESTION 2:

Explanation: Normal server client chat, which doesnt end till BYEBYE message is sent. Looping through the send and receive and checking BYEBYE condition to break out of the loop.

CLIENT:

```
// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080


int main()
{
   int sock = 0, valread;
   struct sockaddr_in serv_addr;
   if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
   {
       printf("\n Socket creation error \n");
       return -1;
   }

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_port = htons(PORT);

   // Convert IPv4 and IPv6 addresses from text to binary form
   if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
   {
       printf("\nInvalid address/ Address not supported \n");
       return -1;
   }

   if (connect(sock, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0)
   {
       printf("\nConnection Failed \n");
       return -1;
   }

   while(1)
   {
   char *str;
   char buffer[1024] = {0};
   printf("Client: ");
   scanf("%[^\n]%*c", str);
   send(sock , str , strlen(str) , 0 );
   valread = read( sock , buffer, sizeof(buffer));
   //check for byebye
   printf("Server: %s\n",buffer );
   if(strcmp(buffer, "BYEBYE") == 0)
```

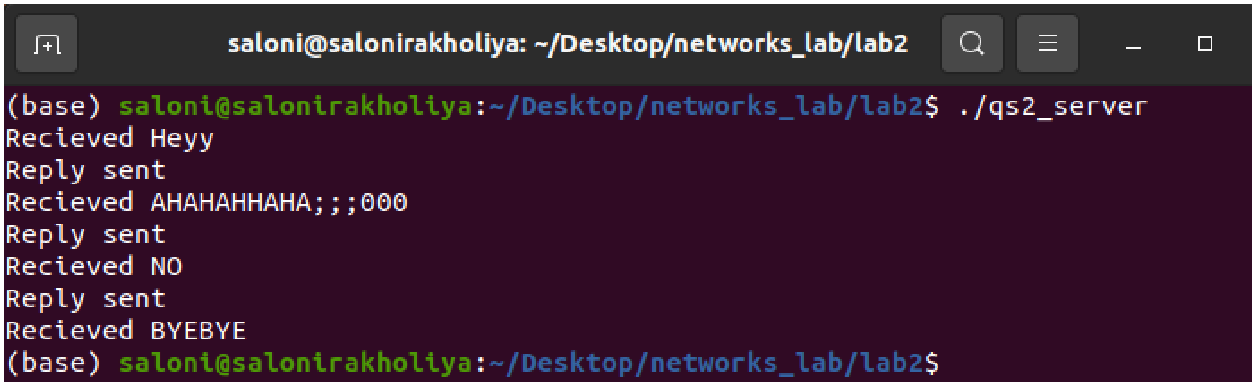
```
      break;
    }
    return 0;
}
```

SERVER:

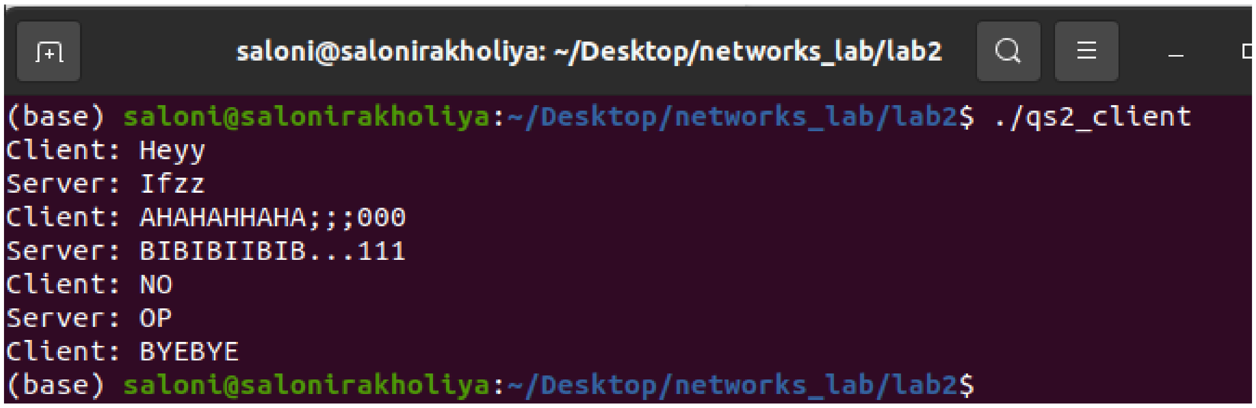
```

// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);


    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                  &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                       (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }

    while(1)
    {
    char buffer[1024] = {0};
    valread = read( new_socket , buffer, 1024);
    printf("Recieved %s\n",buffer );
    if(strcmp(buffer, "BYEBYE") == 0)
    {
        send(new_socket , buffer , strlen(buffer) , 0 );
        break;
    }
    else {
        //Any character otherthan a letter or a digit will be replaced
by a period(.)”
```

```c
        for(int i=0;i<strlen(buffer);++i)
        {
            if(buffer[i]>='a' && buffer[i]<='z')
            {
                if(buffer[i]=='z') buffer[i]='a';
                else buffer[i]=(char)(buffer[i]+1);
            }
            else if(buffer[i]>='A' && buffer[i]<='Z')
            {
                if(buffer[i]=='Z') buffer[i]='A';
                else buffer[i]=(char)(buffer[i]+1);
            }
            else if(buffer[i]>='0' && buffer[i]<='9')
            {
                if(buffer[i]=='9') buffer[i]='0';
                else buffer[i]=(char)(buffer[i]+1);
            }
            else buffer[i]='.';

        }
        send(new_socket , buffer , strlen(buffer) , 0 );
    }

    printf("Reply sent\n");
    }

    return 0;
}
```

OUTPUTS:
Example 1:

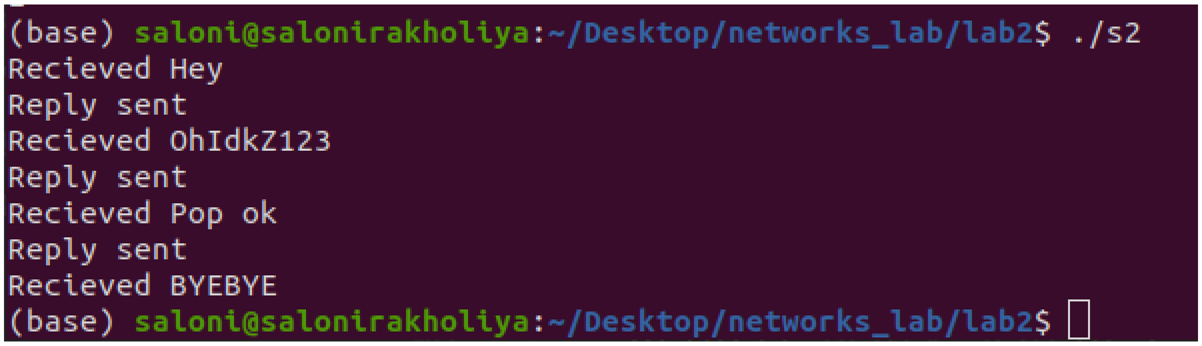


```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs2_server
Recieved Heyy
Reply sent
Recieved AHAHAHHAHA;;;000
Reply sent
Recieved NO
Reply sent
Recieved BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

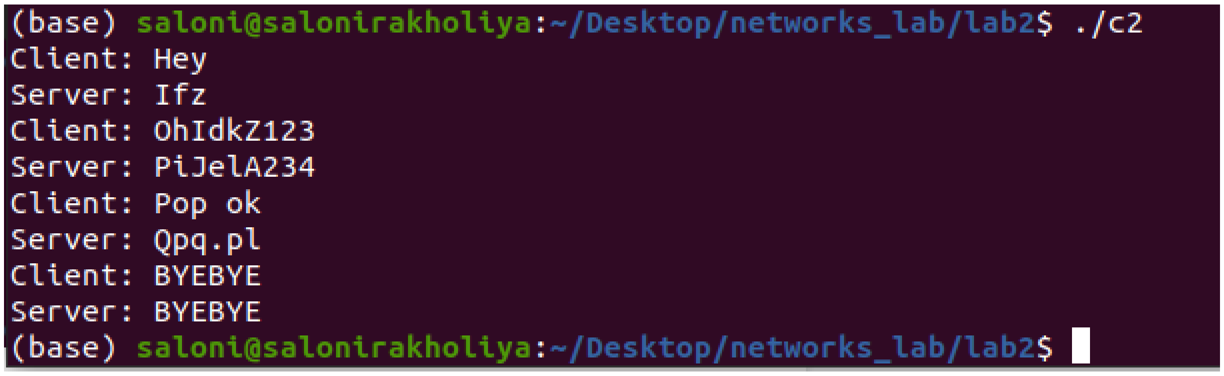


```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./qs2_client
Client: Heyy
Server: Ifzz
Client: AHAHAHHAHA;;;000
Server: BIBIBIIBIB...111
Client: NO
Server: OP
Client: BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

Example 2

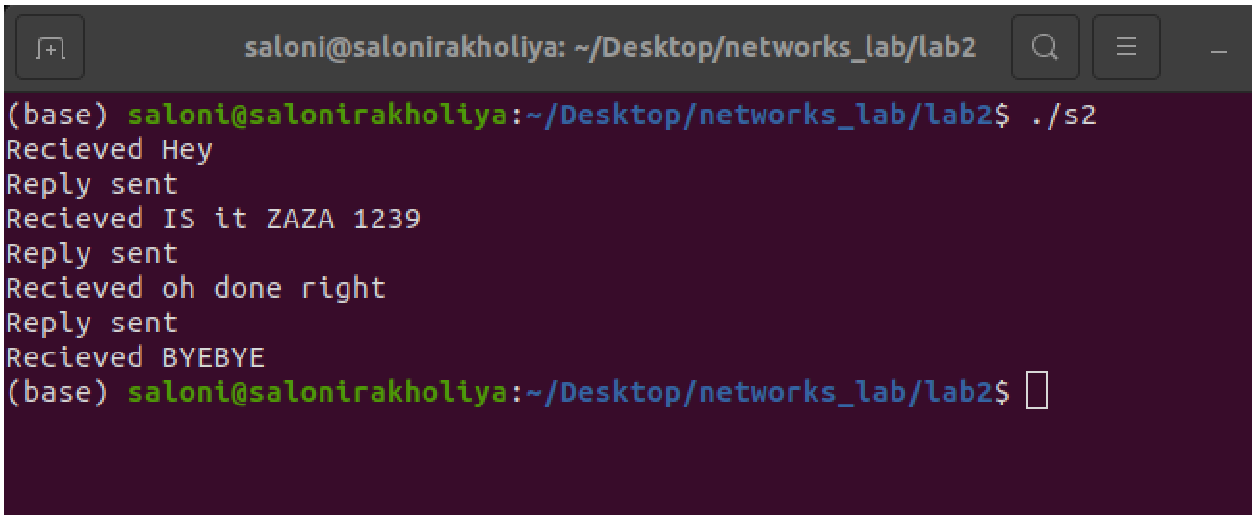


```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./s2
Recieved Hey
Reply sent
Recieved OhIdkZ123
Reply sent
Recieved Pop ok
Reply sent
Recieved BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

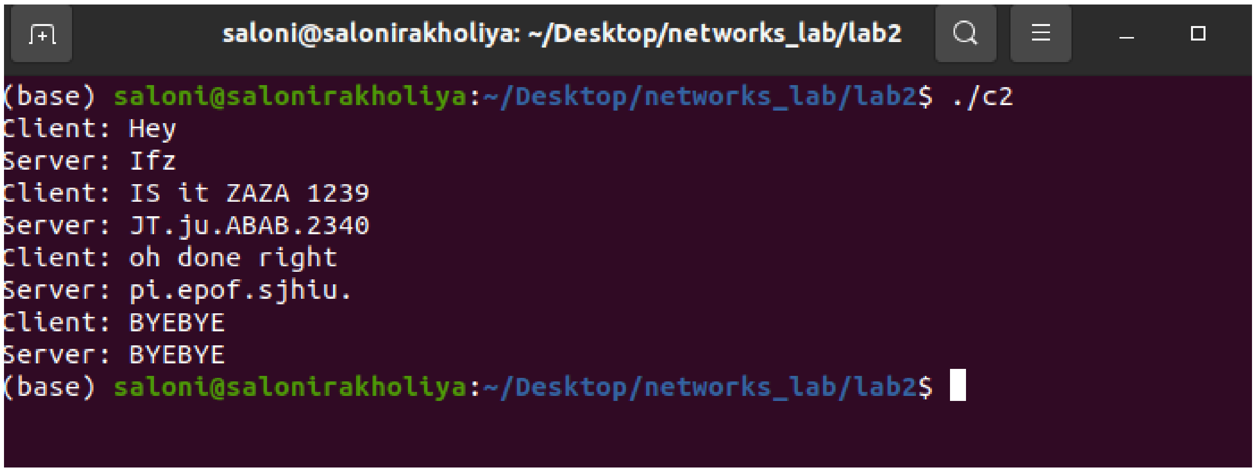


```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./c2
Client: Hey
Server: Ifz
Client: OhIdkZ123
Server: PiJelA234
Client: Pop ok
Server: Qpq.pl
Client: BYEBYE
Server: BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

Example 3:



```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./s2
Recieved Hey
Reply sent
Recieved IS it ZAZA 1239
Reply sent
Recieved oh done right
Reply sent
Recieved BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ 
```



```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./c2
Client: Hey
Server: Ifz
Client: IS it ZAZA 1239
Server: JT.ju.ABAB.2340
Client: oh done right
Server: pi.epof.sjhiu.
Client: BYEBYE
Server: BYEBYE
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ 
```

QUESTION 3:

Explanation: Making a 10MB file using a python script, and using C (commented code in server side) breaking the file into 10 chunks and randomly generating 5 chunks to send. Sending the 5 chunks and the client asks for 5 remaining chunks from serverB. Server B sends remaining chunks and all chunks are also stored as files, which can be combined to form the previous 10 MB File.

CLIENT CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024
```

```c
void write_file(int sockfd, char *filename)
{
    int n = -1;
    FILE *fp;
    char buffer[SIZE];

    fp = fopen(filename, "a");
    char ch;
    while (1)
    {
        n = recv(sockfd, &ch, sizeof(ch), 0);
        if (n <= 0)
            break;
        fprintf(fp, "%c", ch);
        if (ch == EOF)
            return;
    }
    return;
}

int main()
{
    char *ip = "127.0.0.1";
    int port = 8086;
    int e;

    int sockfd;
    struct sockaddr_in server_addr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        perror("[-]Error in socket");
        exit(1);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = port;
    server_addr.sin_addr.s_addr = inet_addr(ip);

    e = connect(sockfd, (struct sockaddr *)&server_addr,
sizeof(server_addr));
    if (e == -1)
    {
        perror("[-]Error in socket");
        exit(1);
    }

    int buffer[10];
    char *msgask = "Send packets!\n";
    send(sockfd, msgask, strlen(msgask), 0);
    printf("%s", msgask);
    int valread = recv(sockfd, buffer, sizeof(buffer), 0);
    printf("Chunks which will be got from server A(1) and which from
B(-1)\n");
    for (int i = 0; i < 10; ++i)
        printf("%d ", buffer[i]);
    //
    printf("\n");
    char *filenames[10] = {"rfile_part1", "rfile_part2",
"rfile_part3", "rfile_part4", "rfile_part5", "rfile_part6",
"rfile_part7", "rfile_part8", "rfile_part9", "rfile_part10"};

    for (int i = 0; i < 10; ++i)
    {
        if (buffer[i] != -1)
        {

            write_file(sockfd, filenames[i]);
            printf("Chunk %d got from server\n", i + 1);
        }
    }
```

```
  //second

      int sockfd2;
      struct sockaddr_in server_addr2;
      int port2=8080;
      sockfd2 = socket(AF_INET, SOCK_STREAM, 0);
      if (sockfd2 < 0)
      {
          perror("[-]Error in socket");
          exit(1);
      }
      server_addr2.sin_family = AF_INET;
      server_addr2.sin_port = port2;
      server_addr2.sin_addr.s_addr = inet_addr(ip);

      e = connect(sockfd2, (struct sockaddr *)&server_addr2,
sizeof(server_addr2));
      if (e == -1)
      {
          perror("[-]Error in socket");
          exit(1);
      }

  send(sockfd2, buffer, sizeof(buffer), 0);

      for(int i=0;i<10;++i)
    {
      if(buffer[i]==-1)
      {

        write_file(sockfd2,filenames[i]);
        printf("Chunk %d got from server\n",i+1);
      }
    }
    char thanks[1024] = "Thank You from client!";
    send(sockfd, thanks, 1024, 0);
    send(sockfd2, thanks,1024, 0);
    return 0;
}
```

SERVER A CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

void send_file(FILE *fp, int sockfd)
{
    int n;
    char data[SIZE] = {0};
    char ch;

    while ((ch = fgetc(fp)) != EOF)
    {
        if (send(sockfd, &ch, sizeof(ch), 0) == -1)
        {
            perror("[-]Error in sending file.");
            exit(1);
        }
    }
    ch = EOF;
    if (send(sockfd, &ch, sizeof(ch), 0) == -1)
    {
        perror("[-]Error in sending file.");
        exit(1);
    }
}

FILE *openforwrite(int filecounter)
{
```

```c
    char fileoutputname[15];

    sprintf(fileoutputname, "file_part%d", filecounter);
    return fopen(fileoutputname, "a");
}

int main()
{
    //preprocess file
    // FILE *ptr_readfile;
    // FILE *ptr_writefile;
    // char line [128];
    // int filecounter=1, linecounter=1;

    // ptr_readfile = fopen("randomTxt2.txt","r");
    // if (!ptr_readfile)
    //  return 1;

    // ptr_writefile = openforwrite(filecounter);

    // while (fgets(line, sizeof line, ptr_readfile)!=NULL) {
    //  if (linecounter == 11) {
    //      linecounter = 1;
    //      filecounter=1;
    //  }
    //  ptr_writefile = openforwrite(filecounter);
    //  if (!ptr_writefile)
    //      return 1;
    //  fprintf(ptr_writefile,"%s\n", line);
    //  fclose(ptr_writefile);
    //  linecounter++;
    //  filecounter++;
    // }
    // fclose(ptr_readfile);
    //preprocessing ends
    //deciding chunks
    int total = 0;
    int a[10];
    for (int i = 0; i < 10; ++i)
        a[i] = -1;

    while (1)
    {
        int x = rand() % 10;
        if (a[x] == -1)
        {
            total += 1;
            a[x] = 1;
        }
        if (total == 5)
            break;
    }
    printf("Checker to track which chunks are sent from server A(1) and which are not(-1)\n");
    for (int i = 0; i < 10; ++i)
        printf("%d", a[i]);

    printf("\n");

    char *ip = "127.0.0.1";
    int port = 8086;
    int e;

    int sockfd, new_sock;
    struct sockaddr_in server_addr, new_addr;
    socklen_t addr_size;
    char buffer[SIZE];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        perror("[-]Error in socket");
        exit(1);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = port;
    server_addr.sin_addr.s_addr = inet_addr(ip);

    e = bind(sockfd, (struct sockaddr *)&server_addr,
sizeof(server_addr));
    if (e < 0)
    {
        perror("[-]Error in bind");
        exit(1);
    }
    if (listen(sockfd, 10) == 0)
    {
        printf("[+]Listening....\n");
    }
    else
    {
        perror("[-]Error in listening");
        exit(1);
    }

    addr_size = sizeof(new_addr);
    new_sock = accept(sockfd, (struct sockaddr *)&new_addr,
&addr_size);

    char *qs;

    int valread = recv(new_sock, qs, 1024, 0);
    send(new_sock, a, sizeof(a), 0);

    char *filenames[10] = {"file_part1", "file_part2", "file_part3",
"file_part4", "file_part5", "file_part6", "file_part7", "file_part8",
"file_part9", "file_part10"};
```

```
    for (int i = 0; i < 10; ++i)
    {
        if (a[i] != -1)
        {

            FILE *fp;
            //char *filename = "randomTxt2.txt";
            fp = fopen(filenames[i], "r");
            if (fp == NULL)
            {
                perror("[-]Error in reading file.");
                exit(1);
            }
            send_file(fp, new_sock);
            printf("Chunk %d sent from server\n", i + 1);
        }
    }

    char waitmsgthanks[1024];
    while (1)
    {
        valread = recv(new_sock, waitmsgthanks, 1024, 0);
        if (valread <= 0)
            break;
    }
    printf("%s", waitmsgthanks);
    return 0;
}
```

SERVER B CODE:

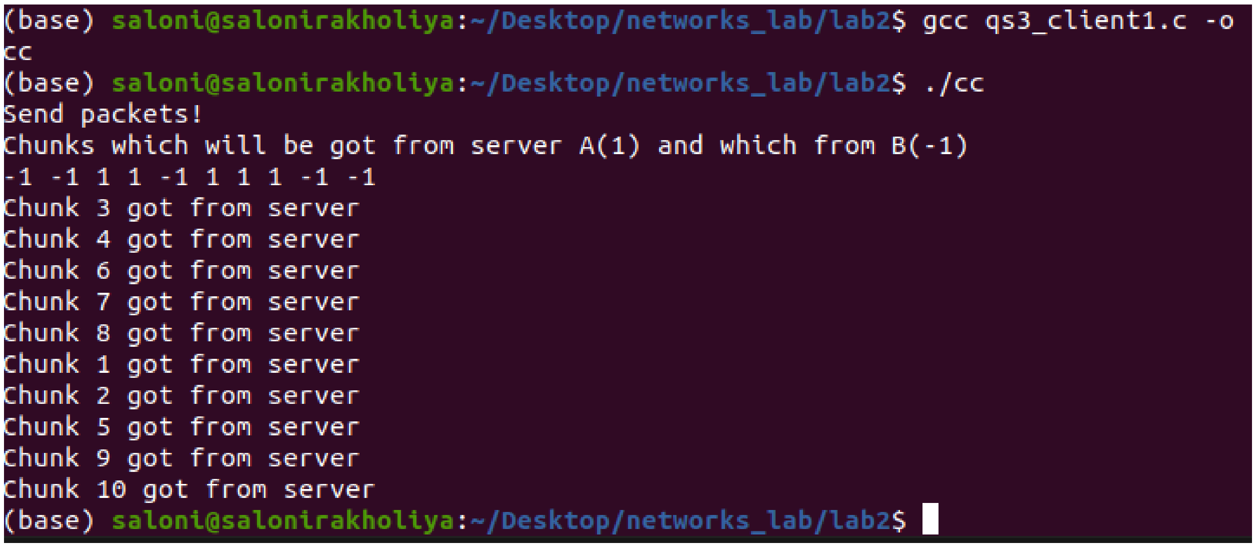
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

void send_file(FILE *fp, int sockfd)
```

```c
{
  int n;
  char data[SIZE] = {0};
  char ch;
  while ((ch = fgetc(fp)) != EOF)
  {
    if (send(sockfd, &ch, sizeof(ch), 0) == -1)
    {
      perror("[-]Error in sending file.");
      exit(1);
    }
  }
  ch = EOF;
  if (send(sockfd, &ch, sizeof(ch), 0) == -1)
  {
    perror("[-]Error in sending file.");
    exit(1);
  }
}

int main()
{
  char *ip = "127.0.0.1";
  int port = 8080;
  int e;

  int sockfd, new_sock;
  struct sockaddr_in server_addr, new_addr;
  socklen_t addr_size;
  char buffer[SIZE];

  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0)
  {
    perror("[-]Error in socket");
    exit(1);
```

```
 }
 printf("[+]Server socket created successfully.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = port;
 server_addr.sin_addr.s_addr = inet_addr(ip);

 e = bind(sockfd, (struct sockaddr *)&server_addr,
sizeof(server_addr));
 if (e < 0)
 {
   perror("[-]Error in bind");
   exit(1);
 }
 printf("[+]Binding successfull.\n");

 if (listen(sockfd, 10) == 0)
 {
   printf("[+]Listening....\n");
 }
 else
 {
   perror("[-]Error in listening");
   exit(1);
 }

 addr_size = sizeof(new_addr);
 new_sock = accept(sockfd, (struct sockaddr *)&new_addr, &addr_size);

 int a[10];
 char *filenames[10] = {"file_part1", "file_part2", "file_part3",
"file_part4", "file_part5", "file_part6", "file_part7", "file_part8",
"file_part9", "file_part10"};

 int valread = recv(new_sock, a, sizeof(a), 0);
 printf("Chunks to send from server B(-1)\n");
```

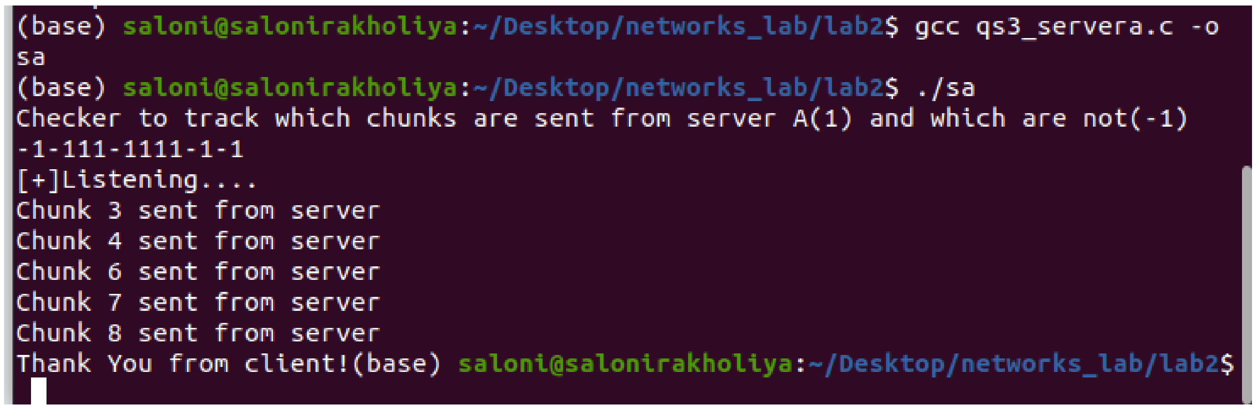
```
  for (int i = 0; i < 10; ++i)
    printf("%d", a[i]);
  printf("\n");
  for (int i = 0; i < 10; ++i)
  {
    if (a[i] == -1)
    {

      FILE *fp;
      fp = fopen(filenames[i], "r");
      if (fp == NULL)
      {
        perror("[-]Error in reading file.");
        exit(1);
      }
      send_file(fp, new_sock);
      printf("Chunk %d sent from server\n", i + 1);
    }
  }

  char waitmsgthanks[1024];
  valread = recv(new_sock, waitmsgthanks, 1024, 0);
  printf("%s", waitmsgthanks);
  return 0;
}
```

OUTPUT:
Client side



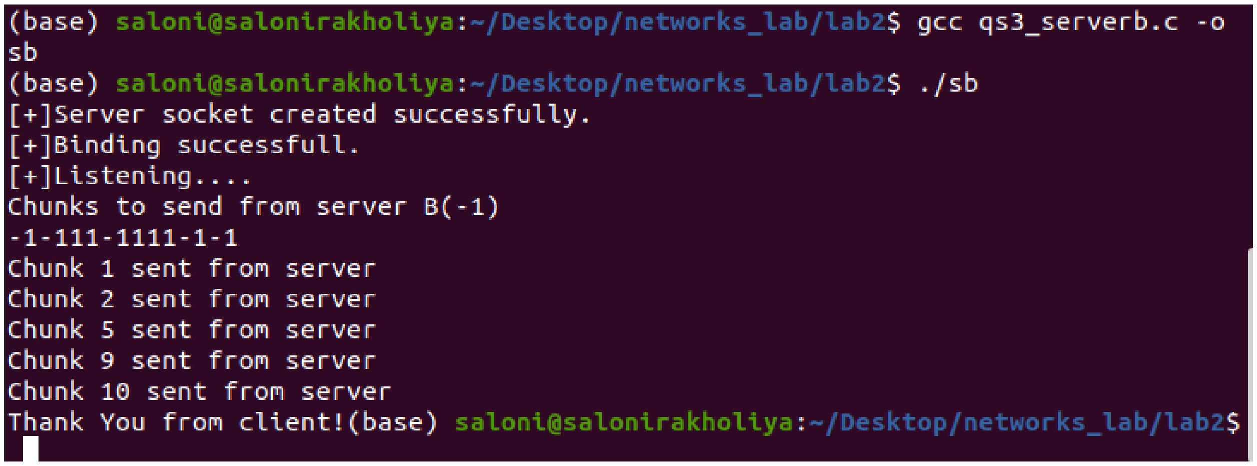
```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs3_client1.c -o
cc
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./cc
Send packets!
Chunks which will be got from server A(1) and which from B(-1)
-1 -1 1 1 -1 1 1 1 -1 -1
Chunk 3 got from server
Chunk 4 got from server
Chunk 6 got from server
Chunk 7 got from server
Chunk 8 got from server
Chunk 1 got from server
Chunk 2 got from server
Chunk 5 got from server
Chunk 9 got from server
Chunk 10 got from server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

Server A side:



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs3_servera.c -o
sa
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./sa
Checker to track which chunks are sent from server A(1) and which are not(-1)
-1-111-1111-1-1
[+]Listening....
Chunk 3 sent from server
Chunk 4 sent from server
Chunk 6 sent from server
Chunk 7 sent from server
Chunk 8 sent from server
Thank You from client!(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```

Server B side:



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ gcc qs3_serverb.c -o
sb
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$ ./sb
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
Chunks to send from server B(-1)
-1-111-1111-1-1
Chunk 1 sent from server
Chunk 2 sent from server
Chunk 5 sent from server
Chunk 9 sent from server
Chunk 10 sent from server
Thank You from client!(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab2$
```