NETWORKS LAB

ROLL NUMBER: **106119109**

NAME: **SALONI RAKHOLIYA**

Week 1: (12-08-2021) Socket Programming

QUESTION 1: (a):

CODE:

SERVER CODE:

```
//SERVER
#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()
{
   struct sockaddr_in address;
   int server_fd, new_socket, valread;
   int opt = 1;
   int addrlen = sizeof(address);
   char ch;
   // Creating socket file descriptor
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
   {
       perror("socket failed");
       exit(EXIT_FAILURE);
   }

   //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 );

   // 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);
   }
   valread = read( new_socket , &ch, 1024);
   printf("character recieved from client 1: %c\n",ch );
   ch-=1;
   //FIRST DONE

   // 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);
   }

   send(new_socket , &ch , 1 , 0 );
   printf("Decremented Character sent to client 2\n");

   return 0;
}
```

CLIENT 1 CODE:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main()
{
   int mysocket = 0;
```

```
    struct sockaddr_in serv_addr;
    char ch;
    if ((mysocket = 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(mysocket, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
< 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    printf("Enter character to send to server: ");
    scanf("%c", &ch);
    send(mysocket , &ch , 1, 0 );
    printf("Character message sent from client 1 to server\n");
    return 0;
}
```

CLIENT 2 CODE:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080
```

```c
int main()
{
   int mysocket = 0, val;
   struct sockaddr_in serv_addr;
   if ((mysocket = 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(mysocket, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
< 0)
   {
       printf("\nConnection Failed \n");
       return -1;
   }
   printf("Character recieved from server to client2\n");
   char ch;
   val = read( mysocket , &ch, 1024);
   printf("%c\n",ch );
   return 0;
}
```

SCREENSHOTS:

```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1a_server.c -o server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
character recieved from client 1: i
Decremented Character sent to client 2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1a_client.c -o client1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1a_client2.c -o client2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client1
Enter character to send to server: i
Character message sent from client 1 to server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client2
Character recieved from server to client2
h
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

QUESTION 1: (b):

CODE:

SERVER CODE:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <math.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main()
{
   int serverfiledesc, new_socket, valread;
   struct sockaddr_in address;
   int opt = 1;
   int addrlen = sizeof(address);
   float float_val;
   // Creating socket file desc
   if ((serverfiledesc = socket(AF_INET, SOCK_STREAM, 0)) == 0)
   {
       perror("socket failed");
       exit(EXIT_FAILURE);
   }

   // attaching socket to the port 8080
   if (setsockopt(serverfiledesc, 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(serverfiledesc, (struct sockaddr *)&address,
sizeof(address))<0)
   {
       perror("bind failed");
       exit(EXIT_FAILURE);
   }
   if (listen(serverfiledesc, 3) < 0)
   {
       perror("listen");
       exit(EXIT_FAILURE);
   }
   if ((new_socket = accept(serverfiledesc, (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
   {
       perror("accept");
       exit(EXIT_FAILURE);
   }
   valread = read( new_socket , &float_val, 1024);
   printf("Float value recieved from client 1: %f\n",float_val );
   float_val=pow(float_val,1.5);
   //FIRST DONE

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

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

   send(new_socket , &float_val ,sizeof(float_val), 0 );
   printf("Increased float by power 1.5 sent to client 2\n");

   return 0;
}
```

CLIENT 1 CODE:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main()
{
```

```
    int mysocket = 0;
    struct sockaddr_in serv_addr;
    float float_val;
    if ((mysocket = 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(mysocket, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
< 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    printf("Enter Float value to send to server: ");
    scanf("%f", &float_val);
    send(mysocket , &float_val , sizeof(float_val), 0 );
    printf("Float sent from client 1 to server\n");
    return 0;
}
```

CLIENT 2 CODE:

```
#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, val;
   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;
   }
   printf("Float recieved from server to client2\n");
   float float_val;
   val = read( sock , &float_val, 1024);
   printf("%f\n",float_val );
   return 0;
}
```

SCREENSHOTS:

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_server.c -o server -l
m
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
Float value recieved from client 1: 6.990000
Increased float by power 1.5 sent to client 2
```

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_client1.c -o client1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_client2.c -o client2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client1
Enter Float value to send to server: 6.99
Float sent from client 1 to server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client2
Float recieved from server to client2
18.480587
```

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_server.c -o server -l
m
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
Float value recieved from client 1: 6.990000
Increased float by power 1.5 sent to client 2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_client1.c -o client1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1b_client2.c -o client2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client1
Enter Float value to send to server: 6.99
Float sent from client 1 to server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client2
Float recieved from server to client2
18.480587
```

QUESTION 1: (c):

CODE:

SERVER CODE:

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080

struct mystruct{
   int i;
   float f;
   char ch;
};

int main()
{
   int server_fd, new_socket, valread;
   struct sockaddr_in address;
```

```
int opt = 1;
int addrlen = sizeof(address);
struct mystruct currstruct;
struct mystruct sendingstruct={10,10.5,'y'};
// Creating socket file descr
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
    perror("socket failed");
    exit(EXIT_FAILURE);
}

// 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);
}
```

```
   valread = read( new_socket , &currstruct, 1024);
   printf("struct recieved from client 1: \nchar: %c \nint: %d, \nfloat:
%f\n",currstruct.ch, currstruct.i, currstruct.f );

   //FIRST DONE

   // 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);
    }

    send(new_socket , &sendingstruct , sizeof(struct mystruct) , 0 );
    printf("New struct sent to client 2\n");

    return 0;
}
```

CLIENT 1 CODE:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080
//struct to send
struct mystruct{
    int i;
    float f;
    char ch;
};

int main()
{
    int mysocket = 0;
    struct sockaddr_in serv_addr;
    struct mystruct currstruct={6,2.5,'x'};
    if ((mysocket = 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)
```

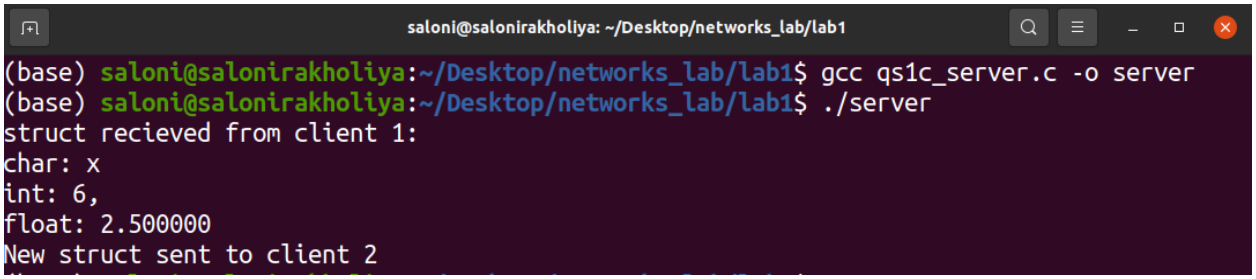
```
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(mysocket, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
< 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    printf("Sending struct to server with: \nchar %c:  \nint %d , \nfloat
%f",currstruct.ch,currstruct.i,currstruct.f);
    send(mysocket , &currstruct , sizeof(struct mystruct), 0 );
    printf("\nStruct sent from client 1 to server\n");
    return 0;
}
```

CLIENT 2 CODE:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

struct mystruct{
    int i;
    float f;
    char ch;
};

int main()
{
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    struct mystruct currstruct;
    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;
    }

    valread = read( sock , &currstruct, 1024);
    printf("Struct recieved from server to client2\n");
    printf("struct recieved: \nchar: %c \nint: %d \nfloat:
%f\n",currstruct.ch, currstruct.i, currstruct.f );
    return 0;
}
```

SCREENSHOTS:



```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1c_client1.c -o client1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs1c_client2.c -o client2
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client1
Sending struct to server with:
char x:
int 6 ,
float 2.500000
Struct sent from client 1 to server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client2
Struct recieved from server to client2
struct recieved:
char: y
int: 10
float: 10.500000
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

QUESTION 2:

CODE:

SERVER CODE:

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

using namespace std;

typedef struct {
 int number;
 char name[10];
 int price;
 char description[10];
 int quantity;
 int subParts[2];
} partStruct;

typedef struct {
```

```
 int partno;
 int quantity;
} orderStruct;

typedef struct {
 int acno;
 orderStruct ord;
} customerStruct;

partStruct parts[5];
customerStruct customers[5];

void HANDLE_SEND_RECV_ERRORS(int st) {
 if ((st) == -1) {
   perror("Error in send()/recv()");
   exit(254);
 } else if ((st) == 0) {
   perror("Connection is closed because send/recv returned 0");
   exit(255);
 }
}

void fill_db() {
 strcpy(parts[0].name, "Wheel");
 strcpy(parts[0].description, "Car Wheel");
 parts[0].number = 0;
 parts[0].price = 5000;
 parts[0].quantity = 100;
 parts[0].subParts[0] = 1; // tyre
 parts[0].subParts[1] = -1;

 strcpy(parts[1].name, "Tyre");
 strcpy(parts[1].description, "Rubber Tyre");
 parts[1].number = 1;
 parts[1].price = 2000;
 parts[1].quantity = 200;
 parts[1].subParts[0] = -1;
 parts[1].subParts[1] = -1;

 strcpy(parts[2].name, "Door");
```

```
 strcpy(parts[2].description, "Metal Door");
 parts[2].number = 2;
 parts[2].price = 10000;
 parts[2].quantity = 150;
 parts[2].subParts[0] = -1;
 parts[2].subParts[1] = -1;

 strcpy(parts[3].name, "Engine");
 strcpy(parts[3].description, "Petrol Engine");
 parts[3].number = 3;
 parts[3].price = 50000;
 parts[3].quantity = 20;
 parts[3].subParts[0] = 4; // shaft
 parts[3].subParts[1] = -1;

 strcpy(parts[4].name, "Shaft");
 strcpy(parts[4].description, "Crankshaft");
 parts[4].number = 4;
 parts[4].price = 7500;
 parts[4].quantity = 50;
 parts[4].subParts[0] = -1;
 parts[4].subParts[1] = -1;

 customers[0].acno = 0;
 customers[1].acno = 1;
 customers[2].acno = 2;
 customers[3].acno = 3;
 customers[4].acno = 4;
}

void die_with_error(const char *errorMessage) {
 perror(errorMessage);
 exit(1);
}

int recv_int(int sock) {
 int res = -1;
 int recv_msg_size;

 if ((recv_msg_size = recv(sock, &res, sizeof(res), 0)) < 0)
```

```
  die_with_error("recv() failed");
 return res;
}

void send_str(int sock, char *msg_buf) {
 int status = send(sock, msg_buf, strlen(msg_buf), 0);
 HANDLE_SEND_RECV_ERRORS(status);
 if (status != strlen(msg_buf))
   die_with_error("Couldn't send() fully");
}

void send_int(int sock, int payload) {
 int status = send(sock, &payload, sizeof(payload), 0);
 HANDLE_SEND_RECV_ERRORS(status);
 if (status != sizeof(payload))
   die_with_error("Couldn't send() fully");
}

void recv_str(int sock, char buf[30]) {
 int recv_msg_size;

 recv_msg_size = recv(sock, buf, 30, 0);
 HANDLE_SEND_RECV_ERRORS(recv_msg_size);
}

void handle_client(int client_socket, int queryno) {
 if (queryno < 0 || queryno > 5)
   return;
 printf("Recieved query number %d\n", queryno);

 if (queryno == 1) {
   char name[10];
   strcpy(name, "Invalid");
   int partno = recv_int(client_socket);

   if (partno < 5)
     strcpy(name, parts[partno].name);

   send_str(client_socket, name);
 } else if (queryno == 2) {
```

```
  int partno = recv_int(client_socket);
  int result = 0;

  if (partno < 5)
    result = parts[partno].quantity;

  send_int(client_socket, result);
} else if (queryno == 3) {
  int custid = recv_int(client_socket);
  int partno = recv_int(client_socket);
  int quantity = recv_int(client_socket);

  char response[30];
  strcpy(response, "Failed");

  if (partno < 5 && parts[partno].quantity >= quantity) {
    if (custid < 5) {
      customers[custid].ord.partno = partno;
      customers[custid].ord.quantity = quantity;

      parts[partno].quantity -= quantity;

      strcpy(response, "Success");
    }
  }

  send_str(client_socket, response);
} else if (queryno == 4) {
  int partno = recv_int(client_socket);
  char response[30];
  strcpy(response, "Empty");

  if (partno < 5) {
    int num = 0;
    for (int i = 0; i < 5 && parts[partno].subParts[i] != -1; ++i)
      ++num;

    send_int(client_socket, num);

    for (int i = 0; i < num; ++i) {
```

```
      strcpy(response, parts[parts[partno].subParts[i]].name);
      send_str(client_socket, response);
    }
  }
 } else if (queryno == 5) {
  char partname[30];
  recv_str(client_socket, partname);

  int partno = -1;
  for (int i = 0; i < 5; ++i)
    if (strcmp(partname, parts[i].name) == 0)
      partno = i;

  send_int(client_socket, partno);
 }
}

int main() {
 int server_socket;
 int client_socket;
 struct sockaddr_storage client_addr;
 unsigned int client_len;


 fill_db();

 struct addrinfo hints, *res;
 memset(&hints, 0, sizeof(hints));
 hints.ai_family = AF_UNSPEC;
 hints.ai_socktype = SOCK_STREAM;
 hints.ai_flags = AI_PASSIVE;

 if (getaddrinfo(NULL, to_string(8080).c_str(), &hints, &res) == -1) {
  die_with_error("getaddrinfo()");
 }

 if ((server_socket =
         socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0)
  die_with_error("socket() failed");
```

```
 // Bind to the local address
 if (bind(server_socket, res->ai_addr, res->ai_addrlen) < 0)
   die_with_error("bind() failed");

 // frees heap memory
 freeaddrinfo(res);

 //socket listening
 if (listen(server_socket, 5) < 0)
   die_with_error("listen() failed");

 int queryno = 0;
 while (queryno != -1) {
   client_len = sizeof(client_addr);

   if ((client_socket = accept(server_socket, (struct sockaddr
*)&client_addr,&client_len)) < 0)
     die_with_error("accept() failed");

   while (queryno != -1) {
     queryno = recv_int(client_socket);
     handle_client(client_socket, queryno);
   }
 }

 close(client_socket);

 return 0;
}
```

CLIENT CODE:

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

```
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
using namespace std;

void HANDLE_SEND_RECV_ERRORS(int st) {
 if ((st) == -1) {
   perror("Error in send()/recv()");
   exit(254);
 } else if ((st) == 0) {
   perror("Connection is closed because send/recv returned 0");
   exit(255);
 }
}

void die_with_error(const char *errorMessage) {
 perror(errorMessage);
 exit(1);
}

int recv_int(int sock) {
 int res = -1;
 int recv_msg_size;

 recv_msg_size = recv(sock, &res, sizeof(res), 0);
 HANDLE_SEND_RECV_ERRORS(recv_msg_size);
 return res;
}

void send_str(int sock, char msg_buf[30]) {
 int len = send(sock, msg_buf, strlen(msg_buf), 0);
 HANDLE_SEND_RECV_ERRORS(len);
 if (len != strlen(msg_buf))
   die_with_error("Couldn't send() fully");
}

void send_int(int sock, int payload) {
 int len = send(sock, &payload, sizeof(payload), 0);
 HANDLE_SEND_RECV_ERRORS(len);
 if (len != sizeof(payload))
```

```
   die_with_error("Couldn't send() fully");
}

void recv_str(int sock, char buf[30]) {
 int recv_msg_size;

 recv_msg_size = recv(sock, buf, 30, 0);
 HANDLE_SEND_RECV_ERRORS(recv_msg_size);
 buf[recv_msg_size] = '\0';
}

void handle(int sock, int q_no) {
 if (q_no < 0 || q_no > 5)
   return;

 if (q_no == 1) {
   int partno;
   printf("Enter the part number= ");
   scanf("%d", &partno);

   send_int(sock, partno);
   char partname[30];
   recv_str(sock, partname);

   printf("%s\n\n", partname);
 } else if (q_no == 2) {
   int partno;
   printf("Enter the part number= ");
   scanf("%d", &partno);

   send_int(sock, partno);
   int qty = recv_int(sock);

   printf("part quantity= %d\n\n", qty);
 } else if (q_no == 3) {
   int custid, partno, qty;

   printf("Enter user id= ");
   scanf("%d", &custid);
```

```
  printf("Enter the part number= ");
  scanf("%d", &partno);

  printf("Enter part's quantity= ");
  scanf("%d", &qty);

  send_int(sock, custid);
  send_int(sock, partno);
  send_int(sock, qty);

  char status[30];
  recv_str(sock, status);

  printf("Current status= %s\n\n", status);
} else if (q_no == 4) {
  int partno;
  printf("Enter the part number= ");
  scanf("%d", &partno);

  send_int(sock, partno);
  int num = recv_int(sock);

  printf("%d subparts of given part= \n", num);

  char response[30];
  for (int i = 0; i < num; ++i) {
    recv_str(sock, response);
    printf("%s, ", response);
  }
  printf("\n\n");
} else if (q_no == 5) {
  char partname[30];

  printf("Enter the part name= ");
  scanf("%s", partname);

  send_str(sock, partname);
  int partno = recv_int(sock);

  if (partno == -1)
```

```
    printf("Does not exist\n\n");
  else
    printf("Part number= %d\n\n", partno);
 }
}

int main() {
 int sock;
 unsigned short server_port;
 server_port = 8080;

 struct addrinfo myaddr, *res;
 memset(&myaddr, 0, sizeof(myaddr));
 myaddr.ai_family = AF_UNSPEC;
 myaddr.ai_socktype = SOCK_STREAM;

 int status =
    getaddrinfo("127.0.0.1", to_string(server_port).c_str(), &myaddr,
&res);
 if (status == -1)
   die_with_error("getaddrinfo() failed");

 if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) <
0)
   die_with_error("socket() failed");

 if (connect(sock, res->ai_addr, res->ai_addrlen) < 0)
   die_with_error("connect() failed");

 freeaddrinfo(res);

 int q_no = 0;
 while (q_no != -5) {
   printf("MENU\n");
   printf("========================================\n");
   printf("1. Get part name from number\n2. Get quantity of avaliable
parts\n3. place an order for a part\n4. Get the list of subparts\n5.
search for a part\nEnter query to send: \n");

   scanf("%d", &q_no);
```

```
    send_int(sock, q_no);
    handle(sock, q_no);
  }

  close(sock);
  exit(0);
}
```

SCREENSHOTS:

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ g++ qs2_server.cpp -o server -lm
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
Recieved query number 1
Recieved query number 2
Recieved query number 3
Recieved query number 4
Recieved query number 5
```

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ g++ qs2_client.cpp -o client
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client
MENU
========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
1
Enter the part number= 2
Door

MENU
========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
2
Enter the part number= 1
part quantity= 200
```

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
MENU
==========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
3
Enter user id= 1
Enter the part number= 3
Enter part's quantity= 2
Current status= Success

MENU
==========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
4
Enter the part number= 2
0 subparts of given part=
```

```
MENU
==========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
5
Enter the part name= Wheel
Part number= 0

MENU
==========================================
1. Get part name from number
2. Get quantity of avaliable parts
3. place an order for a part
4. Get the list of subparts
5. search for a part
Enter query to send:
-5
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

QUESTION 3:

CODE:

SERVER:

```
// Server
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
 #define PORT     8080
#define MAXLINE 1024
 // Driver code
int main() {
   int sockfd;
```

```
struct sockaddr_in servaddr, cliaddr;
int flag_done=0;
// Creating socket file descriptor
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
    perror("socket creation failed");
    exit(EXIT_FAILURE);
}

memset(&servaddr, 0, sizeof(servaddr));
memset(&cliaddr, 0, sizeof(cliaddr));

// Filling server information
servaddr.sin_family    = AF_INET; // IPv4
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);

// Bind the socket with the server address
if ( bind(sockfd, (const struct sockaddr *)&servaddr,
        sizeof(servaddr)) < 0 )
{
    perror("bind failed");
    exit(EXIT_FAILURE);
}

int len, n;
 len = sizeof(cliaddr);  //len is value/resuslt
 while(1)
{
float a[5];
float b[5];
n = recvfrom(sockfd, a, MAXLINE,
            MSG_WAITALL, ( struct sockaddr *) &cliaddr,
            &len);

printf("Array 1:\n");
for(int i=0;i<5;++i) printf("%f ",a[i]);

n = recvfrom(sockfd, b, MAXLINE,
            MSG_WAITALL, ( struct sockaddr *) &cliaddr,
            &len);
```

```
   printf("\nArray 2:\n");
   for(int i=0;i<5;++i) printf("%f ",b[i]);

   int flag=0;
   for(int i=0;i<5;++i)
   if(((int)a[i]%2) |||((int)b[i]%2) || (int)floor((double)a[i])!=(int)a[i]
|| (int)floor((double)b[i])!=(int)b[i]) flag=1;

   if(flag==0)
   {
       int x[2];
       int sum1=0;
       int sum2=0;
       for(int i=0;i<5;++i)
       {
           sum1+=a[i];
           sum2+=b[i];
       }
       x[0]=sum1;
       x[1]=sum2;
       sendto(sockfd, x, MAXLINE,
       MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
           len);
       printf("\nSums sent from server to client.\n");
       break;
   }
   else printf("\nArray had error!!\n");
   }
   return 0;
}
```

CLIENT:

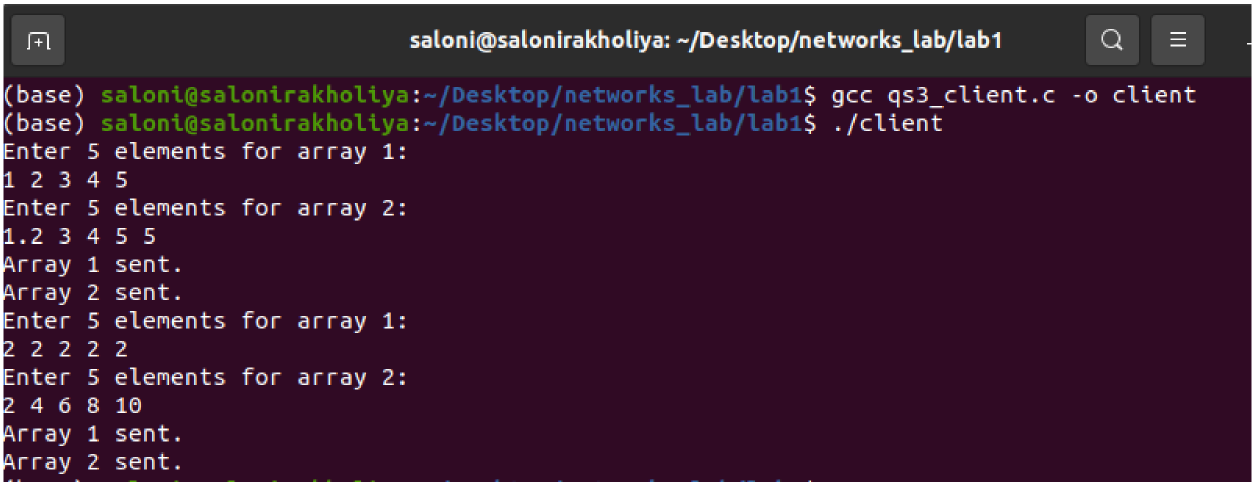
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <arpa/inet.h>
#include <netinet/in.h>
 #define PORT     8080
#define MAXLINE 1024

int main() {
   int socket_desc;
   //timestamp for retransmission
   struct timeval tv;
   tv.tv_sec = 0;
   tv.tv_usec = 100000;
   int flag_done=0;
   float a[5];
   float b[5];
   struct sockaddr_in     servaddr;
    // Creating socket file descriptor
   if ( (socket_desc = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
       perror("socket creation failed");
       exit(EXIT_FAILURE);
   }
    memset(&servaddr, 0, sizeof(servaddr));

   // Filling server information
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(PORT);
   servaddr.sin_addr.s_addr = INADDR_ANY;

   //retransmission timeout

   if (setsockopt (socket_desc, SOL_SOCKET, SO_RCVTIMEO, (char *)&tv,
sizeof(tv)) < 0)
   perror("setsockopt failed\n");
   //
   int n=-1, len;
   while(n<0)
   {
   printf("Enter 5 elements for array 1: \n");
   for(int i=0;i<5;++i)
   scanf("%f",&a[i]);
```

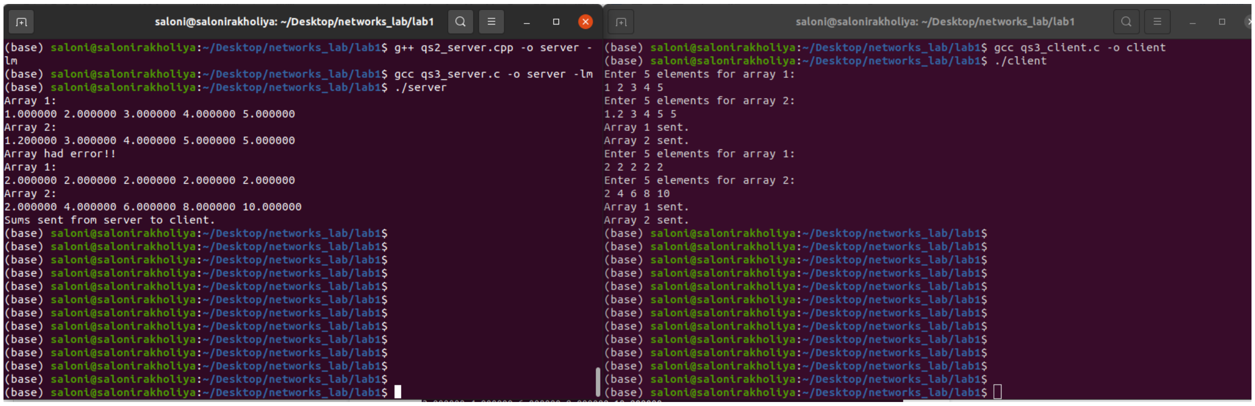
```
   printf("Enter 5 elements for array 2: \n");
   for(int i=0;i<5;++i)
   scanf("%f",&b[i]);

   sendto(socket_desc, a, sizeof(a),
       MSG_CONFIRM, (const struct sockaddr *) &servaddr,
           sizeof(servaddr));
   printf("Array 1 sent.\n");

   sendto(socket_desc, b, sizeof(b),
       MSG_CONFIRM, (const struct sockaddr *) &servaddr,
           sizeof(servaddr));
   printf("Array 2 sent.\n");
   int x[2];
   n = recvfrom(socket_desc, x, sizeof(x), MSG_WAITALL, (struct sockaddr
*) &servaddr,&len);

   if(n>=0)
   for(int i=0;i<2;++i)
   printf("%d ",x[i]);
   }
   //close socket
   close(socket_desc);
   return 0;
}
```

SCREENSHOTS:

```
saloni@salonirakholiya: ~/Desktop/networks_lab/lab1
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ g++ qs2_server.cpp -o server -
lm
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs3_server.c -o server -lm
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
Array 1:
1.000000 2.000000 3.000000 4.000000 5.000000
Array 2:
1.200000 3.000000 4.000000 5.000000 5.000000
Array had error!!
Array 1:
2.000000 2.000000 2.000000 2.000000 2.000000
Array 2:
2.000000 4.000000 6.000000 8.000000 10.000000
Sums sent from server to client.
```



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs3_client.c -o client
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client
Enter 5 elements for array 1:
1 2 3 4 5
Enter 5 elements for array 2:
1.2 3 4 5 5
Array 1 sent.
Array 2 sent.
Enter 5 elements for array 1:
2 2 2 2 2
Enter 5 elements for array 2:
2 4 6 8 10
Array 1 sent.
Array 2 sent.
```



```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ g++ qs2_server.cpp -o server -
lm
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs3_server.c -o server -lm
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./server
Array 1:
1.000000 2.000000 3.000000 4.000000 5.000000
Array 2:
1.200000 3.000000 4.000000 5.000000 5.000000
Array had error!!
Array 1:
2.000000 2.000000 2.000000 2.000000 2.000000
Array 2:
2.000000 4.000000 6.000000 8.000000 10.000000
Sums sent from server to client.
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```

```
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ gcc qs3_client.c -o client
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$ ./client
Enter 5 elements for array 1:
1 2 3 4 5
Enter 5 elements for array 2:
1.2 3 4 5 5
Array 1 sent.
Array 2 sent.
Enter 5 elements for array 1:
2 2 2 2 2
Enter 5 elements for array 2:
2 4 6 8 10
Array 1 sent.
Array 2 sent.
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
(base) saloni@salonirakholiya:~/Desktop/networks_lab/lab1$
```