Expt 3

Client Program

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
int main( int argc, char *argv[])
{
struct sockaddr_in server;
int sd ;
char buffer[200];
if((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
perror("Socket failed:");
exit(1);
}
bzero(&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[2]));
inet_pton(AF_INET, argv[1], &server.sin_addr);
if(connect(sd, (struct sockaddr *)&server, sizeof(server))< 0)
{
perror("Connection failed:");
exit(1);
}
fgets(buffer, sizeof(buffer), stdin);
buffer[strlen(buffer)- 1] = '\0';
write (sd,buffer, sizeof(buffer));
read(sd,buffer, sizeof(buffer));
printf("%s\n", buffer);
close(fd);
}
```

Server Program

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
int main( int argc, char *argv[])
{
struct sockaddr_in server, cli;
int cli_len;
int sd, n, i, len;
int data, temp;
char buffer[100];
if((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
perror("Socket failed:");
exit(1);
}
bzero(&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[1]));
server.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(sd, (struct sockaddr*)&server, sizeof(server)) < 0)
{
perror("bind failed:");
exit(1);
}
listen(sd,5);
if((data = accept(sd , (struct sockaddr *) &cli, &cli_len)) < 0)
{
perror("accept failed:");
exit(1);
}
read(data,buffer, sizeof(buffer));
len = strlen(buffer);
for( i =0; i<= len/2; i++)
{
temp = buffer[i];
buffer[i] = buffer[len- 1-i];
buffer[len-1-i] = temp;
}
write (data,buffer, sizeof(buffer));
close(data);
close(sd);
}
```

Client program

```
 #include<stdio.h>
 #include<string.h>
 #include<sys/socket.h>
 #include<sys/types.h>
 #include<netinet/in.h>
 #include<arpa/inet.h>
 #include<fcntl.h>
 #include<stdlib.h>
 main(int argc, char * argv[])
 {
 int i,j,n;
 int sock_fd;
 struct sockaddr_in servaddr;
int matrix_1[10][10], matrix_2[10][10], matrix_product[10][10];
 int size[2][2];
 int num_rows_1, num_cols_1, num_rows_2, num_cols_2;
 if(argc != 3)
 {
 }
 fprintf(stderr, "Usage: ./client IPaddress_of_server port\n");
 exit(1);
 printf("Enter the number of rows of first matrix\n");
 scanf("%d", &num_rows_1);
 printf("Enter the number of columns of first matrix\n");
 scanf("%d", &num_cols_1);
 printf("Enter the values row by row one on each line\n" );
 for ( i = 0; i < num_rows_1; i++)
 for( j=0; j<num_cols_1; j++)
 {
 scanf("%d", &matrix_1[i][j]);
 }
 size[0][0] = num_rows_1;
 size[0][1] = num_cols_1;
 printf("Enter the number of rows of second matrix\n");
 scanf("%d", &num_rows_2);
 printf("Enter the number of columns of second matrix\n");
 scanf("%d", &num_cols_2);
 if( num_cols_1 != num_rows_2)
 {
 printf("MATRICES CANNOT BE MULTIPLIED\n");
 exit(1);
 }
 printf("Enter the values row by row one on each line\n");
 for (i = 0; i < num_rows_2; i++)
 for(j=0; j<num_cols_2; j++)
{scanf("%d", &matrix_2[i][j]);
```

```
}
size[1][0] = num_rows_2;
size[1][1] = num_cols_2;
if((sock_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
printf("Cannot create socket\n");
exit(1);
}
bzero((char*)&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(atoi(argv[2]));
inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
// SENDING MATRIX WITHSIZES OFMATRICES1AND2
n =sendto(sock_fd, size, sizeof(size),0, (struct sockaddr*)&servaddr, sizeof(servaddr));
if( n < 0)
{
}
perror("error in matrix 1 sending");
exit(1);
// SENDING MATRIX 1
n =sendto(sock_fd, matrix_1, sizeof(matrix_1),0, (struct sockaddr*)&servaddr, sizeof(servaddr));
if( n < 0)
{
perror("error in matrix 1 sending");
exit(1);
}
// SENDING MATRIX 2
n =sendto(sock_fd, matrix_2, sizeof(matrix_2),0, (struct sockaddr*)&servaddr, sizeof(servaddr));
if( n < 0)
{
perror("error in matrix 2 sending");
exit(1);
}
if((n=recvfrom(sock_fd, matrix_product, sizeof(matrix_product),0, NULL, NULL)) ==-1)
{
perror("read error from server:");
exit(1);
}
printf("\n\nTHE PRODUCT OF MATRICES IS \n\n\n");
for( i=0; i < num_rows_1; i++)
{
for( j=0; j<num_cols_2; j++)
{
printf("%d ",matrix_product[i][j]);
}
printf("\n");
}
close(sock_fd);
}
```

Server Program

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<stdlib.h>
main(int argc, char * argv[])
{
int n;
int sock_fd;
int i,j,k;
int row_1, row_2, col_1, col_2;
struct sockaddr_in servaddr, cliaddr;
int len = sizeof(cliaddr);
int matrix_1[10][10], matrix_2[10][10], matrix_product[10][10];
int size[2][2];
if(argc != 2)
{
fprintf(stderr, "Usage: ./server port\n");
exit(1);
}
if((sock_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
printf("Cannot create socket\n");
exit(1);
}
bzero((char*)&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(atoi(argv[1]));
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(sock_fd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0)
{
perror("bind failed:");
exit(1);
}
// MATRICES RECEIVE
if((n = recvfrom(sock_fd, size, sizeof(size), 0, (struct sockaddr *)&cliaddr, &len)) ==-1)
{
perror("size not received:");
exit(1);
}
// RECEIVE MATRIX 1
if((n = recvfrom(sock_fd, matrix_1, sizeof(matrix_1), 0, (struct sockaddr *)&cliaddr, &len)) ==-1)
{
```

```
perror("matrix 1 not received:");
exit(1);
}
// RECEIVE MATRIX 2
if((n = recvfrom(sock_fd, matrix_2, sizeof(matrix_2), 0, (struct sockaddr *)&cliaddr, &len)) ==-1)
{
perror("matrix 2 not received:");
exit(1);
}
row_1 = size[0][0];
col_1 = size[0][1];
row_2 = size[1][0];
col_2 = size[1][1];
for (i =0; i < row_1 ; i++)
for (j =0; j <col_2; j++)
{
matrix_product[i][j] = 0;
}
for(i =0; i< row_1 ; i++)
for(j=0; j< col_2 ; j++)
for (k=0; k < col_1; k++)
{
matrix_product[i][j] += matrix_1[i][k]*matrix_2[k][j];
}
n =sendto(sock_fd, matrix_product, sizeof(matrix_product),0, (struct sockaddr*)&cliaddr,
sizeof(cliaddr));
if( n < 0)
{
perror("error in matrix product sending");
exit(1);
}
close(sock_fd);
}
```