Say Hello Program:

Server Side

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

int main()
{
int welcomeSocket, newSocket;
char buffer[1024];
struct sockaddr_in serverAddr;
struct sockaddr_storage serverStorage;
socklen_t addr_size;
welcomeSocket = socket(PF_INET, SOCK_STREAM, 0);
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

bind(welcomeSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

if(listen(welcomeSocket,5)==0)
printf("Listening\n");
else
printf("Error\n");

addr_size = sizeof serverStorage;
newSocket = accept(welcomeSocket, (struct sockaddr *) &serverStorage, &addr_size);
strcpy(buffer,"Hello World\n");
send(newSocket,buffer,13,0);
```

```
return 0;
}
```

Client Side

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

int main()
{
int clientSocket;
char buffer[1024];
struct sockaddr_in serverAddr;
socklen_t addr_size;
clientSocket = socket(PF_INET, SOCK_STREAM, 0);
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(7891);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

addr_size = sizeof serverAddr;
connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);
recv(clientSocket, buffer, 1024, 0);
printf("Data received: %s",buffer);

return 0;
}
```

****

Output:

Server Output

Listening

Client Output

Data received: Hello World

****

File Transfer Program:

Server Side

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

int main(void)
{
   int listenfd = 0;
   int connfd = 0;
   struct sockaddr_in serv_addr;
   char sendBuff[1024];
   int numrv;

   listenfd = socket(AF_INET, SOCK_STREAM, 0);

   printf("Socket retrieve success\n");
```

```
memset(&serv_addr, '0', sizeof(serv_addr));
memset(sendBuff, '0', sizeof(sendBuff));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(5000);

bind(listenfd, (struct sockaddr*)&serv_addr,sizeof(serv_addr));

if(listen(listenfd, 10) == -1)
{
   printf("Failed to listen\n");
   return -1;
}


while(1)
{
   connfd = accept(listenfd, (struct sockaddr*)NULL ,NULL);

   FILE *fp = fopen("sample_file.txt","rb");
   if(fp==NULL)
   {
      printf("File opern error");
      return 1;
   }
   while(1)
   {
      unsigned char buff[256]={0};
      int nread = fread(buff,1,256,fp);
```

```
        printf("Bytes read %d \n", nread);

        if(nread > 0)
        {
           printf("Sending \n");
           write(connfd, buff, nread);
        }
        if (nread < 256)
        {
           if (feof(fp))
              printf("End of file\n");
           if (ferror(fp))
              printf("Error reading\n");
           break;
        }


     }

     close(connfd);
     sleep(1);
  }


  return 0;
}
```

Client Side

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
```

```
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(void)
{
   int sockfd = 0;
   int bytesReceived = 0;
   char recvBuff[256];
   memset(recvBuff, '0', sizeof(recvBuff));
   struct sockaddr_in serv_addr;

   if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0)
   {
      printf("\n Error : Could not create socket \n");
      return 1;
   }

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_port = htons(5000); // port
   serv_addr.sin_addr.s_addr = inet_addr("172.16.6.168");

   if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<0)
   {
      printf("\n Error : Connect Failed \n");
      return 1;
   }
```

```
   FILE *fp;
   fp = fopen("sample_file.txt", "ab");
   if(NULL == fp)
   {
      printf("Error opening file");
      return 1;
   }
   while((bytesReceived = read(sockfd, recvBuff, 256)) > 0)
   {
      printf("Bytes received %d\n",bytesReceived);
      fwrite(recvBuff, 1,bytesReceived,fp);
   }

   if(bytesReceived < 0)
   {
      printf("\n Read Error \n");
   }


   return 0;
}
```

****

Output:

Server Output

```
Socket retrieve success
Bytes read 0
End of file
```

Client Output

****

Calculator (Arithmetic) Program:

Server Side

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

void main()
{
 int b,sockfd,connfd,sin_size,l,n,len;
 char operator;
 int op1,op2,result;
 if((sockfd=socket(AF_INET,SOCK_STREAM,0))>0)
  printf("socket created sucessfully\n");  //socket creation
 struct sockaddr_in servaddr;
 struct sockaddr_in clientaddr;

 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=6006;

 if((bind(sockfd, (struct sockaddr *)&servaddr,sizeof(servaddr)))==0)
 printf("bind sucessful\n");

 if((listen(sockfd,5))==0) //listen for connections on a socket
  printf("listen sucessful\n");
```

```
sin_size = sizeof(struct sockaddr_in);
if((connfd=accept(sockfd,(struct sockaddr *)&clientaddr,&sin_size))>0);
printf("accept sucessful\n");

read(connfd, &operator,10);
read(connfd,&op1,sizeof(op1));
read(connfd,&op2,sizeof(op2));

switch(operator)
{
 case '+':
  result=op1 + op2;
   printf("Result is: %d + %d = %d\n",op1, op2, result);
   break;
 case '-':
  result=op1 - op2;
        printf("Result is: %d - %d = %d\n",op1, op2, result);
        break;
 case '*':
  result=op1 * op2;
        printf("Result is: %d * %d = %d\n",op1, op2, result);
        break;
 case '/':
  result=op1 / op2;
        printf("Result is: %d / %d = %d\n",op1, op2, result);
        break;
 default:
        printf("ERROR: Unsupported Operation");
   }
write(connfd,&result,sizeof(result));
close(sockfd);
```

}

Client Side

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netinet/in.h>
#include <unistd.h>
#include<string.h>
#include<strings.h>
#include <arpa/inet.h>
//#define buffsize  150
void main()
{
 int b,sockfd,sin_size,con,n,len;
 char operator;
 int op1,op2,result;
 if((sockfd=socket(AF_INET,SOCK_STREAM,0))>0)
 printf("socket created sucessfully\n");
  struct sockaddr_in servaddr;
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=6006;
 sin_size = sizeof(struct sockaddr_in);
 if((con=connect(sockfd,(struct sockaddr *) &servaddr, sin_size))==0); //initiate a connection on a
socket
  printf("connect sucessful\n");
  printf("Enter operation:\n +:Addition \n -: Subtraction \n /: Division \n*:Multiplication \n");
 scanf("%c",&operator);
  printf("Enter operands:\n");
 scanf("%d %d", &op1, &op2);
```

```
 write(sockfd,&operator,10);
 write(sockfd,&op1,sizeof(op1));
 write(sockfd,&op2,sizeof(op2));
 read(sockfd,&result,sizeof(result));
 printf("Operation result from server=%d\n",result);
 close(sockfd);
}
```

****

Output:

Server Output

```
socket created sucessfully
bind sucessful
listen sucessful
accept sucessful
Result is: 10 + 15 = 25
```

Client Output

```
socket created sucessfully
connect sucessful
Enter operation:
 +:Addition
 -: Subtraction
 /: Division
*:Multiplication
+
Enter operands:
10
15
Operation result from server=25
```

****

Calculator (Trignometry) Program:

Server Side

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netinet/in.h>
#include <unistd.h>
#include<string.h>
#include <arpa/inet.h>
#include<math.h>
#define PI 3.14159265
void main()
{
 int b,sockfd,connfd,sin_size,l,n,len;
 char op;
 double angle1;
 double result,val;

 if((sockfd=socket(AF_INET,SOCK_STREAM,0))>0)
  printf("socket created sucessfully\n");  //socket creation

 struct sockaddr_in servaddr;
 struct sockaddr_in clientaddr;

 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=6666;

 if((bind(sockfd, (struct sockaddr *)&servaddr,sizeof(servaddr)))==0)
  printf("bind sucessful\n");
```

```
if((listen(sockfd,5))==0) //listen for connections on a socket
 printf("listen sucessful\n");

sin_size = sizeof(clientaddr);
if((connfd=accept(sockfd,(struct sockaddr *)&clientaddr,&sin_size))>0);
 printf("accept sucessful\n");
val = PI / 180;
read(connfd, &op,1);

read(connfd, &angle1, sizeof(angle1));

switch(op)
{
      case '1':
  result=sin(angle1*val);
        printf("sin(%lf)=%lf ",angle1,result);
        break;
      case '2':
  result=cos(angle1*val);
        printf("cos(%lf) =%lf ",angle1,result);
            break;
      case '3':
  result=tan(angle1*val);
        printf("tan(%lf) = %lf",angle1,result);
            break;
      default:
            printf("ERROR: Unsupported Operation");
   }
  write(connfd,&result,sizeof(result));
close(connfd);
```

```
 close(sockfd);
}
```

Client Side

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netinet/in.h>
#include <unistd.h>
#include<string.h>
#include<strings.h>
#include <arpa/inet.h>
#include<math.h>
//#define buffsize  150
void main()
{
 int b,sockfd,sin_size,con,n,len;
 double angle,result;
 char op;

 if((sockfd=socket(AF_INET,SOCK_STREAM,0))>0)
 printf("socket created sucessfully\n");

 struct sockaddr_in servaddr;

 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=6666;

 sin_size = sizeof(servaddr);
```

```
 if((con=connect(sockfd,(struct sockaddr *) &servaddr, sin_size))==0); //initiate a connection on a
socket

 printf("connect sucessful\n");


 printf("Enter operation:\n 1:sin \n 2:cos\n 3:tan \n ");

 scanf("%c",&op);


 printf("Enter angle in degree:");

 scanf("%lf",&angle);


 write(sockfd,&op,1);

 write(sockfd,&angle,sizeof(angle));

 read(sockfd,&result,sizeof(result));


 printf("\n Operation result from server=%lf\n",result);

 close(sockfd);

}
```

****

Output:

Server Output

```
socket created sucessfully
bind sucessful
listen sucessful
accept sucessful
sin(90.000000)=1.000000
```

Client Output

```
socket created sucessfully
connect sucessful
Enter operation:
 1:sin
 2:cos
 3:tan
 1
Enter angle in degree:90
Operation result from server=1.000000
```

****

Dhruvil Shah 047

F18111051 TE Comp 1

Assignment 5

Q1 What is socket ? Explain different types of socket

Ans Socket is one endpoint of a two way communication link between two programs running on the network.
The different type of sockets are:

- Stream Socket : Used on the delivery side of the network enviornment.
- Datagram Socket : Doesn't give any guaranteed deliver as they work connectionless.
- Raw Sockets : Supports the developer who builds trending communication protocol.
- Packed Sequenced Socket : It doest have any protecting boundaries.
- Hex Socket
- Socket Bit
- Impact Socket
- Spark Plug Socket
- Pass through socket
- Adjustable multisockets

Q2 Differentiate between TCP and UDP

Ans

Transfer Control Protocol (TCP)	User Datagram Protocol (UDP)
1 Connection oriented protocol	1 Datagram Oriented protocol
2 Connection should be established before transmitting the data	No overhead for opening a connection
3 Reliable as it guarantees	Delivery cannot be guaranteed

	delivery		
4	Provides extensive error checking	4	Provides basic error checking
5	Packets arrive in order	5	No sequencing of data
6	Comparatively slower	6	Comparatively faster
7	Retransmission of lost packets is possible	7	No retransmission of lost packets
8	Doesn't support broadcasting	8	Supports broadcasting
eg	Used by : HTTP, FTP	eg	Used by : DNS.

Q#3 Explain FTP

Ans File transfer protocol is a standard internet protocol provided by TCP / IP. Mainly used for transmitting the web pages, files from their creator to the server and also used for downloading ~~file~~ files to computer from server. It is used to encourage the use of remote computers. It transfers the data more reliably and efficiently.

Q4 Steps involved in establishing a socket on the client side and server side

Ans

Socket on client side:

- Create a socket using the socket () function
- Connect the socket to the address of the server using the connect function.
- Send and reciere data by means of the read () and corite() functions.

Socket on server side:

- Create a socket withe the socket () function
- Bind the socket to an address using bind () function.
- Listen for connections with the listen () function.
- Send and reciere data by means of send () and receive () functions.