```
//cserver.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>

using namespace std;

#define PORT 8052
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 sockaddr_in server_addr,client_addr;

 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
  die("SOCKET CREATE ERROR");
 else
  cout<<"Socket Created.";
 bzero((char *)&server_addr,sizeof(server_addr));
 server_addr.sin_family=AF_INET;
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_port=htons(PORT);

 if(bind(sock,(struct sockaddr*)&server_addr,sizeof(server_addr))==-1)
  die("ERROR IN BINDING");

 if(listen(sock,10)<0)
 {
  die("ERROR WHILE LISTENING");
 }
 socklen_t socklen=sizeof(client_addr);

 int newSocket=accept(sock,(struct sockaddr*)&client_addr,&socklen);

 char buffer[256];

 while(1)
 { cout << "Awaiting client response..." << endl;
  bzero(buffer,256);
  recv(newSocket,buffer,255,0);
  cout<<"Client: "<<buffer<<endl;
  cout<<">";
  //cin.ignore();
  string data;
  getline(cin,data);
  bzero(buffer,256);
```

```
 strcpy(buffer,data.c_str());

 //cin.clear();

    fflush(stdin);
 send(newSocket,(char*)&buffer,strlen(buffer),0);
 }
 return 0;
}



//cclient.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
using namespace std;

#define PORT 8052
#define SERVER_ADDRESS "127.0.0.1"
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 struct sockaddr_in server_addr;
 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
            cout<<"Socket Could Not Be Created";
       else
            cout<<"Socket Created Succesfully";
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_family=AF_INET;
 server_addr.sin_port=htons(PORT);

 int status=connect(sock,(struct sockaddr *)&server_addr,sizeof(server_addr));
  if(status==0)
  cout<<"\nCONNECT SUCCESS!.";
  else
   die("connect");

  char buffer[256];
  while(1)
  {
   bzero((char *)buffer,256);
    cout<<">";
    string data;
    getline(cin, data);
    strcpy(buffer,data.c_str());
```

```
  send(sock,buffer,strlen(buffer),0);
  bzero(buffer,256);
  cout << "Awaiting server response..." << endl;
  recv(sock,(char*)&buffer,sizeof(buffer),0);
  cout<<"Server: "<<buffer<<endl;

 }
return 0;
}


//fserver.cpp


#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<fstream>
using namespace std;

#define PORT 8566
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 sockaddr_in server_addr,client_addr;

 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
  die("SOCKET CREATE ERROR");
 else
  cout<<"Socket Created.";
 bzero((char *)&server_addr,sizeof(server_addr));
 server_addr.sin_family=AF_INET;
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_port=htons(PORT);

 if(bind(sock,(struct sockaddr*)&server_addr,sizeof(server_addr))==-1)
  die("ERROR IN BINDING");

 if(listen(sock,10)<0)
 {
  die("ERROR WHILE LISTENING");
 }
 socklen_t socklen=sizeof(client_addr);

 int newSocket=accept(sock,(struct sockaddr*)&client_addr,&socklen);
 if(newSocket<0)
```

```
 die("ACCEPT ERROR");
else
 cout<<"\nCONNECTION ACCEPTED";
long long int msg_len;


{
 cout<<"\nENter Filename:";
 char filename[100];
 cin>>filename;
 cout<<filename;
 fstream fout;



 msg_len=send(newSocket,filename,100,0); //send filename
 if(msg_len==-1)
  die("Filename error");

 fout.open(filename,ios::in|ios::out|ios::binary);
   fout.seekg(0,ios::end);
   long long int filesize=fout.tellg(); //get file size
   char *filebuff=new char[filesize];
   fout.seekg(0,ios::beg);
   fout.read(filebuff,filesize); //reading file content



 msg_len=send(newSocket,filebuff,filesize,0); //send file conetents
 if(msg_len==-1)
  die("File transmission error");
 else
  cout<<"Transmission Successful";
 fout.close();
}
return 0;
}


//fclient.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<fstream>
using namespace std;
#define SERVER_ADDRESS "127.0.0.1"
#define PORT 8566
void die(char *error)
{
 perror(error);
 exit(1);
```

```
}
int main() {

    struct sockaddr_in server_addr;
  int sock=socket(AF_INET,SOCK_STREAM,0);
  if(sock<0)
                cout<<"Socket Could Not Be Created";
          else
                cout<<"Socket Created Succesfully";
  server_addr.sin_addr.s_addr=INADDR_ANY;
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(PORT);

  int status=connect(sock,(struct sockaddr *)&server_addr,sizeof(server_addr));
   if(status==0)
   cout<<"\nCONNECT SUCCESS!.";
   else
    die("connect");
long long int msg_len;
   char buffer[256];

 {
    cout<<"Wating for server to send  filename.";
    char filename[100];
    bzero((char *)filename,sizeof(filename));
    msg_len=recv(sock,filename,100,0);
    if(msg_len==-1)
     die("Filename error");
    cout<<"\nFilename:"<<filename;


    char *filebuff=new char[90000*80];


    bzero((char *)filebuff,sizeof(filebuff));
    msg_len=recv(sock,filebuff,90000*80,0);
    ofstream fout;
    fout.open(filename,ios::out|ios::binary);
    if(!fout)
     die("CANNOT CREATE FILE");
    else
    {
     fout.write(filebuff,msg_len);
     fout.close();
     cout<<"File received";
    }
   }
 return 0;
}



//server.java
```

```
#include<iostream>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<stdlib.h>  //for exit
#include<unistd.h>
#include <string.h>
#include <stdio.h>
using namespace std;

int main(int argc, char const *argv[])
{
 int sock = socket(AF_INET,SOCK_STREAM,0);
 struct sockaddr_in server,client;

 server.sin_family = AF_INET;
 server.sin_port   = htons(8003);
 server.sin_addr.s_addr = INADDR_ANY;

 if(bind(sock,(struct sockaddr*)&server,sizeof(server))){
  cout<<"\nBIND ERROR\n";
 }

 if (listen(sock,5)<0){
  cout<<"\nERROR ON LISTEN\n";
 }

 socklen_t clientlen = sizeof(client);
 int newsock = accept(sock,(struct sockaddr *)&client,&clientlen);

 float number1,number2,answer;
 char _operator[2],num1[20],num2[20];

 while(1){
  recv(newsock,num1,20,0);
  cout<<"\nThe first number is "<<num1<<endl;
  number1 = atof(num1);
  bzero((char*)num1,sizeof(num1));

  recv(newsock,num2,20,0);
  cout<<"\nThe second number is "<<num2<<endl;
  number2 = atof(num2);
  bzero((char*)num2,sizeof(num2));

  recv(newsock,_operator,2,0);
  cout<<"\nThe operator  is "<<_operator<<endl;

  switch(_operator[0]) {
   case '+':
   {
    char ans[20];
    answer = number1 + number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
```

```
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '-':
   {
    char ans[20];
    answer = number1 - number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '*':
   {
    char ans[20];
    answer = number1 * number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '/':
   {
    char ans[20];
    answer = number1 / number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
  }
  bzero((char*)_operator,sizeof(_operator));
 }
}

// client.cpp

#include<iostream>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>  //for exit
#include<unistd.h>
#include<arpa/inet.h>//for close
using namespace std;

int main()
{
 int n;
 char a[20],b[20],c[20],ans[20];

 int sock = socket(AF_INET,SOCK_STREAM,0);
 struct sockaddr_in server;
```

```
server.sin_family = AF_INET;
server.sin_port   = htons(8003);
server.sin_addr.s_addr = INADDR_ANY;
cout<<ntohl(server.sin_addr.s_addr);

connect(sock,(struct sockaddr *)&server,sizeof(server));

while(1){

 cout<<"\nEnter First Number\n";
 cin>>a;
 send(sock,a,strlen(a),0);
 bzero((char*)a,sizeof(a));

 cout<<"\nEnter Second Number\n";
 cin>>b;
 send(sock,b,strlen(b),0);
 bzero((char*)b,sizeof(b));

 cout<<"\nEnter Operator\n";
 cin>>c;
 send(sock,c,strlen(c),0);
 bzero((char*)c,sizeof(c));

 recv(sock,ans,20,0);
 cout<<"Result:"<<(float)atof(ans);
 bzero((char*)ans,sizeof(ans));
 }
}
```