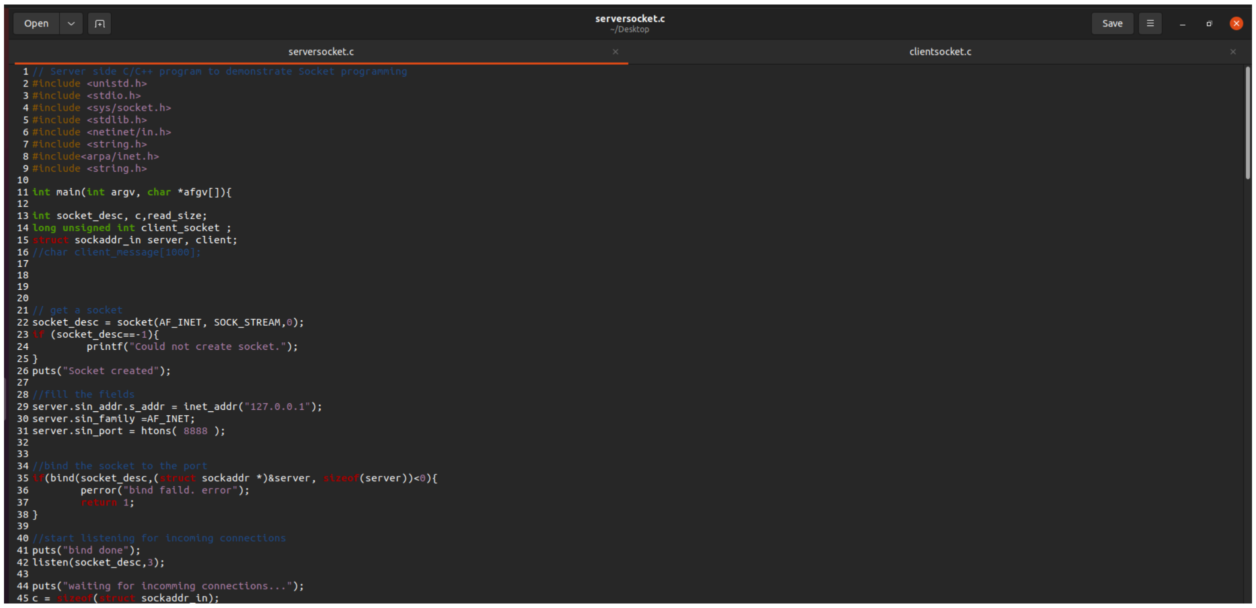
Yash Patel 100746810

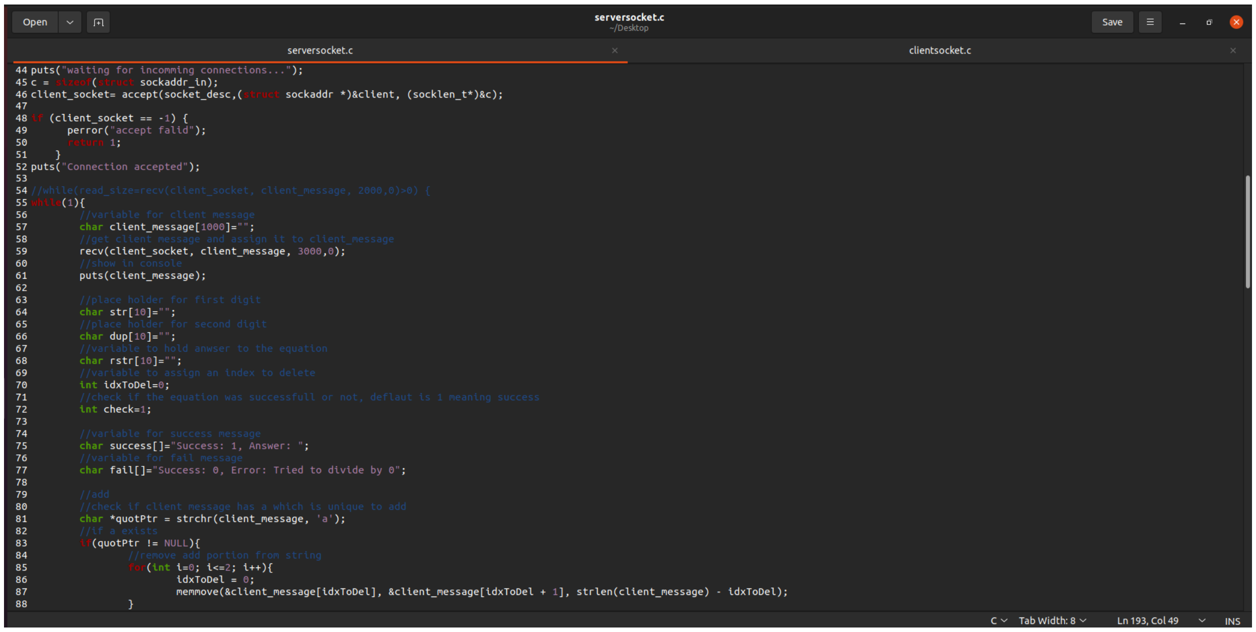
Assignment 2

TCP:

Server

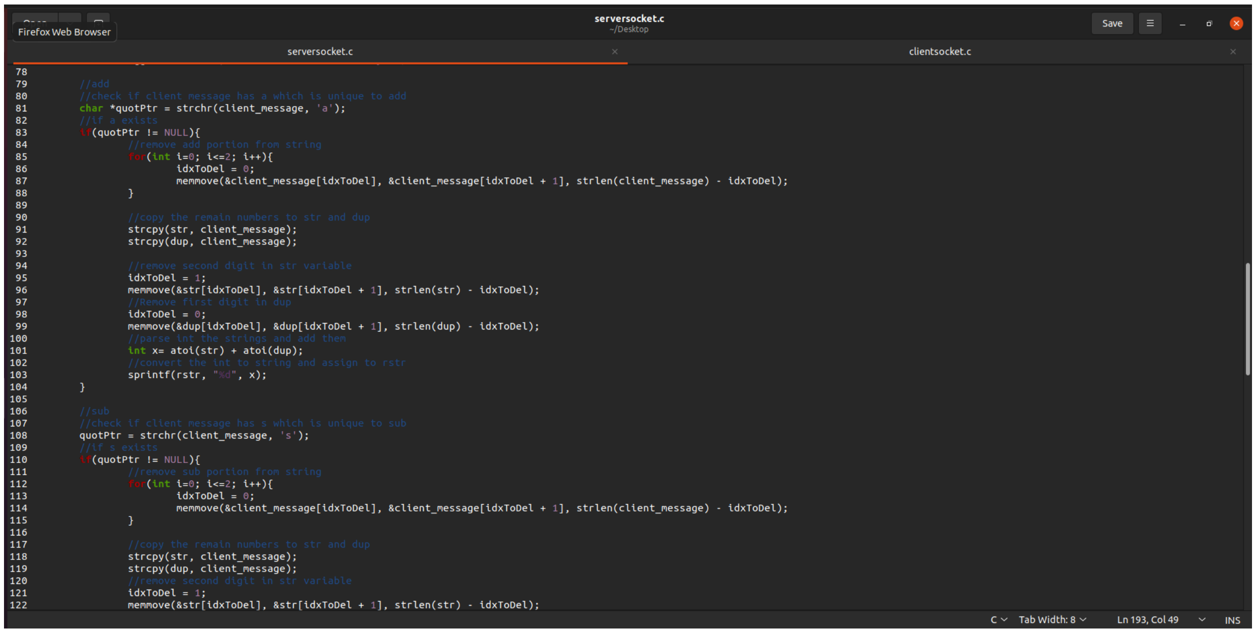


```c
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include<arpa/inet.h>
#include <string.h>

int main(int argv, char *afgv[]){

int socket_desc, c,read_size;
long unsigned int client_socket ;
struct sockaddr_in server, client;
//char client_message[1000];




// get a socket
socket_desc = socket(AF_INET, SOCK_STREAM,0);
if (socket_desc==-1){
	 printf("Could not create socket.");
}
puts("Socket created");

//fill the fields
server.sin_addr.s_addr = inet_addr("127.0.0.1");
server.sin_family =AF_INET;
server.sin_port = htons( 8888 );


//bind the socket to the port
if(bind(socket_desc,(struct sockaddr *)&server, sizeof(server))<0){
	perror("bind faild. error");
	return 1;
}

//start listening for incoming connections
puts("bind done");
listen(socket_desc,3);

puts("waiting for incomming connections...");
c = sizeof(struct sockaddr_in);
```

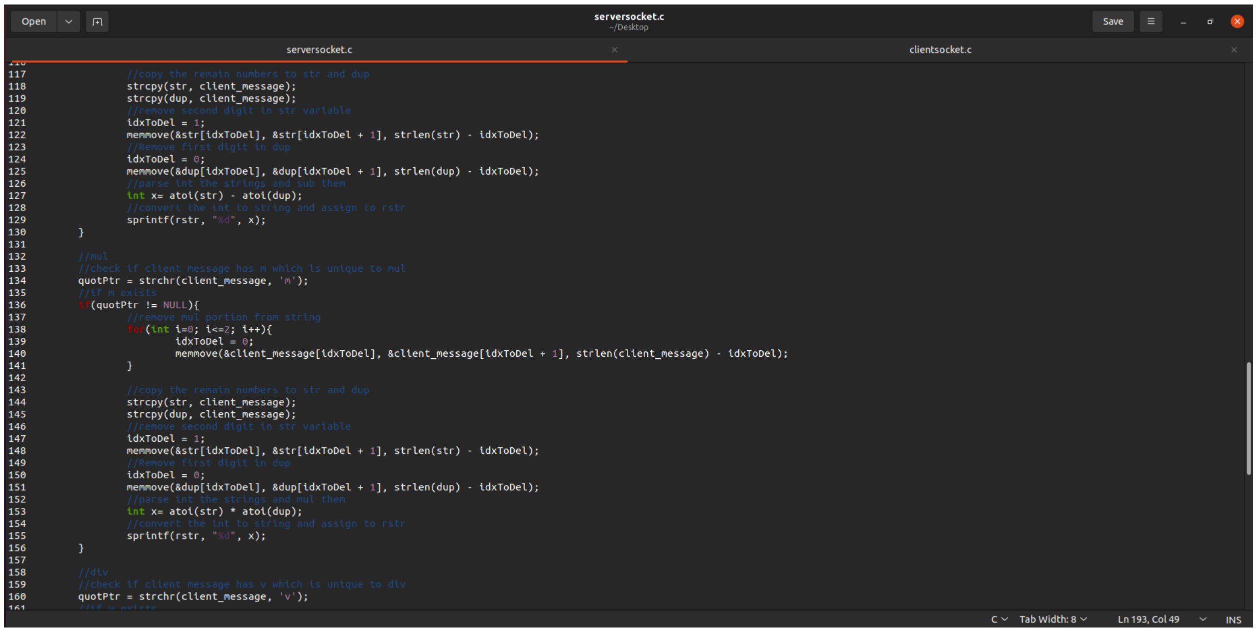


```c
puts("waiting for incomming connections...");
c = sizeof(struct sockaddr_in);
client_socket= accept(socket_desc,(struct sockaddr *)&client, (socklen_t*)&c);

if (client_socket == -1) {
      perror("accept falid");
      return 1;
    }
puts("Connection accepted");

//while(read_size=recv(client_socket, client_message, 2000,0)>0) {
while(1){
	//variable for client message
	char client_message[1000]="";
	//get client message and assign it to client_message
	recv(client_socket, client_message, 3000,0);
	//show in console
	puts(client_message);

	//place holder for first digit
	char str[10]="";
	//place holder for second digit
	char dup[10]="";
	//variable to hold anwser to the equation
	char rstr[10]="";
	//variable to assign an index to delete
	int idxToDel=0;
	//check if the equation was successfull or not, deflaut is 1 meaning success
	int check=1;

	//variable for success message
	char success[]="Success: 1, Answer: ";
	//variable for fail message
	char fail[]="Success: 0, Error: Tried to divide by 0";

	//add
	//check if client message has a which is unique to add
	char *quotPtr = strchr(client_message, 'a');
	//if a exists
	if(quotPtr != NULL){
		//remove add portion from string
		for(int i=0; i<=2; i++){
			idxToDel = 0;
			memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
		}
```

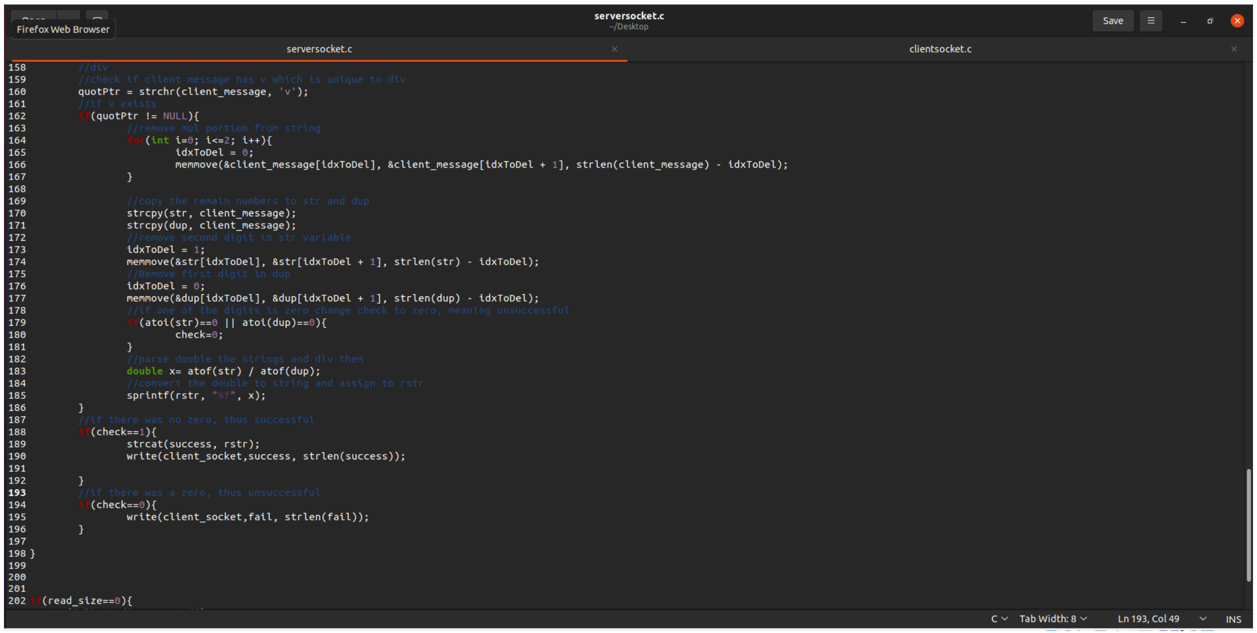
Yash Patel 100746810



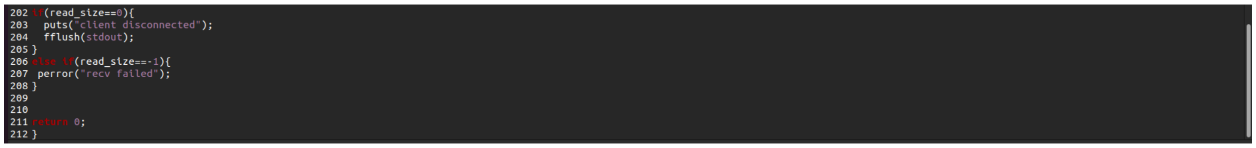
```c

        //add
        //check if client message has a which is unique to add
        char *quotPtr = strchr(client_message, 'a');
        //if a exists
        if(quotPtr != NULL){
                //remove add portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);

                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and add them
                int x= atoi(str) + atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //sub
        //check if client message has s which is unique to sub
        quotPtr = strchr(client_message, 's');
        //if s exists
        if(quotPtr != NULL){
                //remove sub portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
```



```c
                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and sub them
                int x= atoi(str) - atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //mul
        //check if client message has m which is unique to mul
        quotPtr = strchr(client_message, 'm');
        //if m exists
        if(quotPtr != NULL){
                //remove mul portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and mul them
                int x= atoi(str) * atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //div
        //check if client message has v which is unique to div
        quotPtr = strchr(client_message, 'v');
```

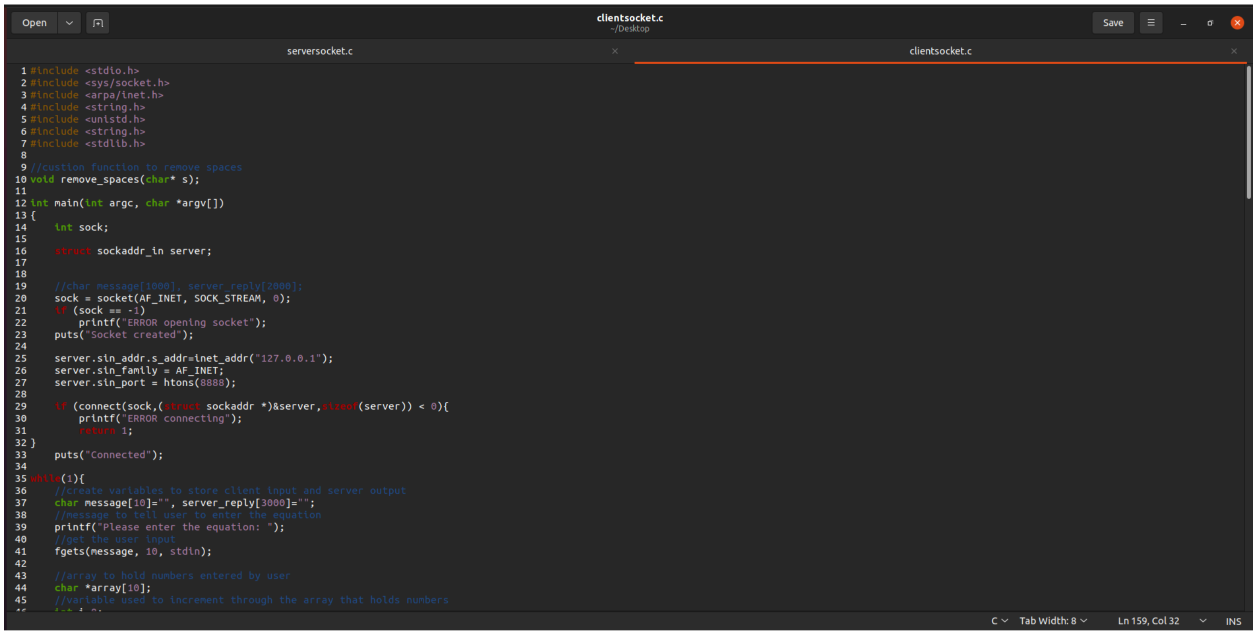


```c
        //div
        //check if client message has v which is unique to div
        quotPtr = strchr(client_message, 'v');
        //if v exists
        if(quotPtr != NULL){
                //remove mul portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //if one of the digits is zero change check to zero, meaning unsuccessful
                if(atoi(str)==0 || atoi(dup)==0){
                        check=0;
                }
                //parse double the strings and div them
                double x= atof(str) / atof(dup);
                //convert the double to string and assign to rstr
                sprintf(rstr, "%f", x);
        }
        //if there was no zero, thus successful
        if(check==1){
                strcat(success, rstr);
                write(client_socket,success, strlen(success));

        }
        //if there was a zero, thus unsuccessful
        if(check==0){
                write(client_socket,fail, strlen(fail));
        }

}



if(read_size==0){
```



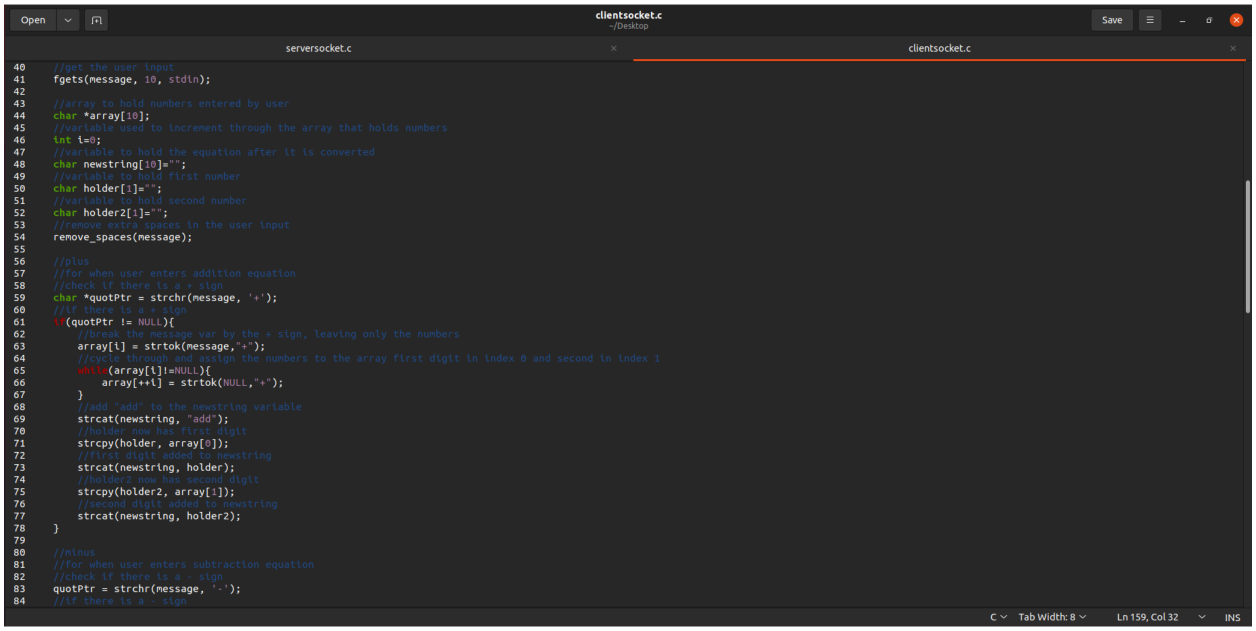
```c
if(read_size==0){
  puts("client disconnected");
  fflush(stdout);
}
else if(read_size==-1){
 perror("recv failed");
}


return 0;
}
```

Client

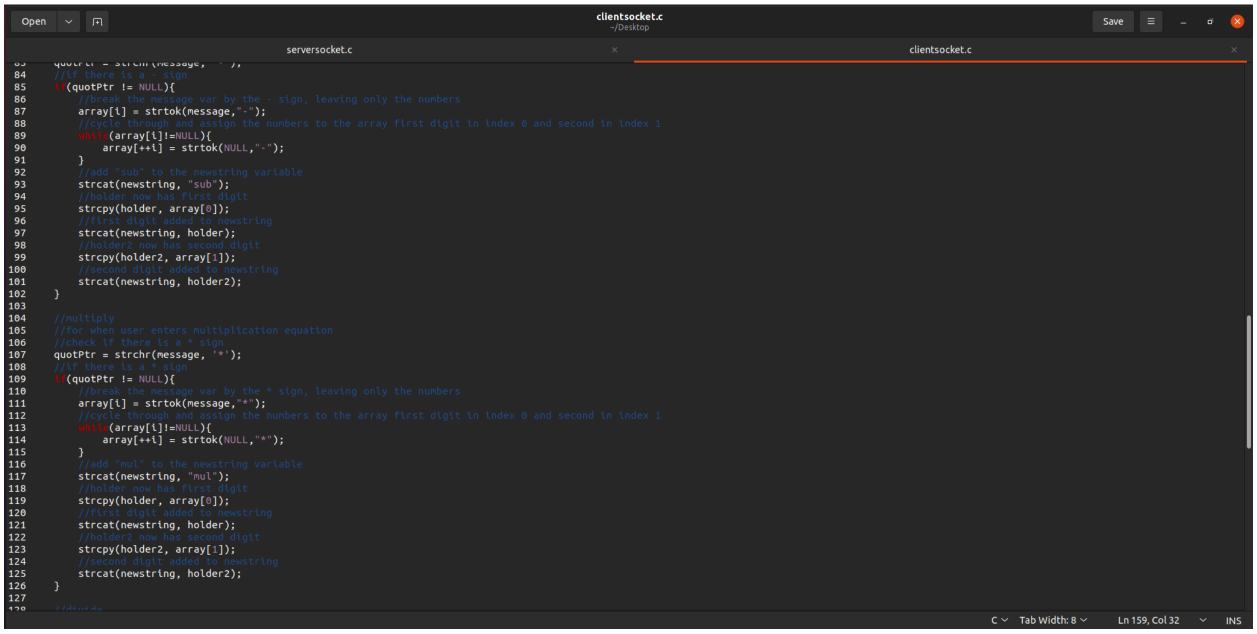


```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

//custion function to remove spaces
void remove_spaces(char* s);

int main(int argc, char *argv[])
{
    int sock;

    struct sockaddr_in server;


    //char message[1000], server_reply[2000];
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1)
        printf("ERROR opening socket");
    puts("Socket created");

    server.sin_addr.s_addr=inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons(8888);

    if (connect(sock,(struct sockaddr *)&server,sizeof(server)) < 0){
        printf("ERROR connecting");
        return 1;
}
    puts("Connected");

while(1){
    //create variables to store client input and server output
    char message[10]="", server_reply[3000]="";
    //message to tell user to enter the equation
    printf("Please enter the equation: ");
    //get the user input
    fgets(message, 10, stdin);

    //array to hold numbers entered by user
    char *array[10];
    //variable used to increment through the array that holds numbers
```

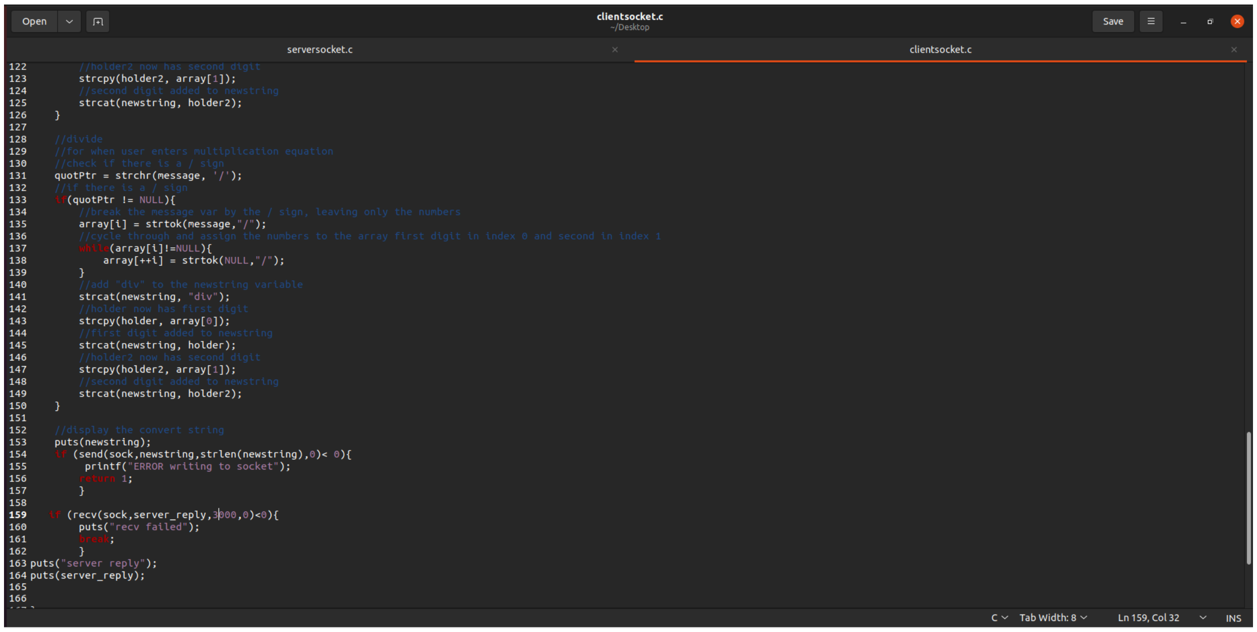
Yash Patel 100746810



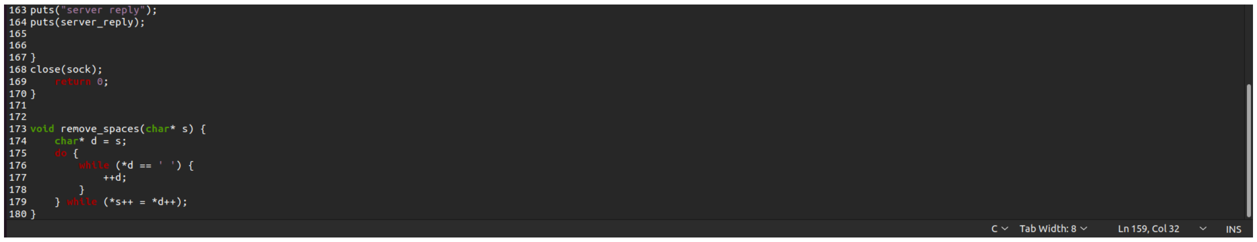
```c
    //get the user input
    fgets(message, 10, stdin);

    //array to hold numbers entered by user
    char *array[10];
    //variable used to increment through the array that holds numbers
    int i=0;
    //variable to hold the equation after it is converted
    char newstring[10]="";
    //variable to hold first number
    char holder[1]="";
    //variable to hold second number
    char holder2[1]="";
    //remove extra spaces in the user input
    remove_spaces(message);

    //plus
    //for when user enters addition equation
    //check if there is a + sign
    char *quotPtr = strchr(message, '+');
    //if there is a + sign
    if(quotPtr != NULL){
        //break the message var by the + sign, leaving only the numbers
        array[i] = strtok(message,"+");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"+");
        }
        //add "add" to the newstring variable
        strcat(newstring, "add");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //minus
    //for when user enters subtraction equation
    //check if there is a - sign
    quotPtr = strchr(message, '-');
    //if there is a - sign
```



```c
    //if there is a - sign
    if(quotPtr != NULL){
        //break the message var by the - sign, leaving only the numbers
        array[i] = strtok(message,"-");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"-");
        }
        //add "sub" to the newstring variable
        strcat(newstring, "sub");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //multiply
    //for when user enters multiplication equation
    //check if there is a * sign
    quotPtr = strchr(message, '*');
    //if there is a * sign
    if(quotPtr != NULL){
        //break the message var by the * sign, leaving only the numbers
        array[i] = strtok(message,"*");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"*");
        }
        //add "mul" to the newstring variable
        strcat(newstring, "mul");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

```

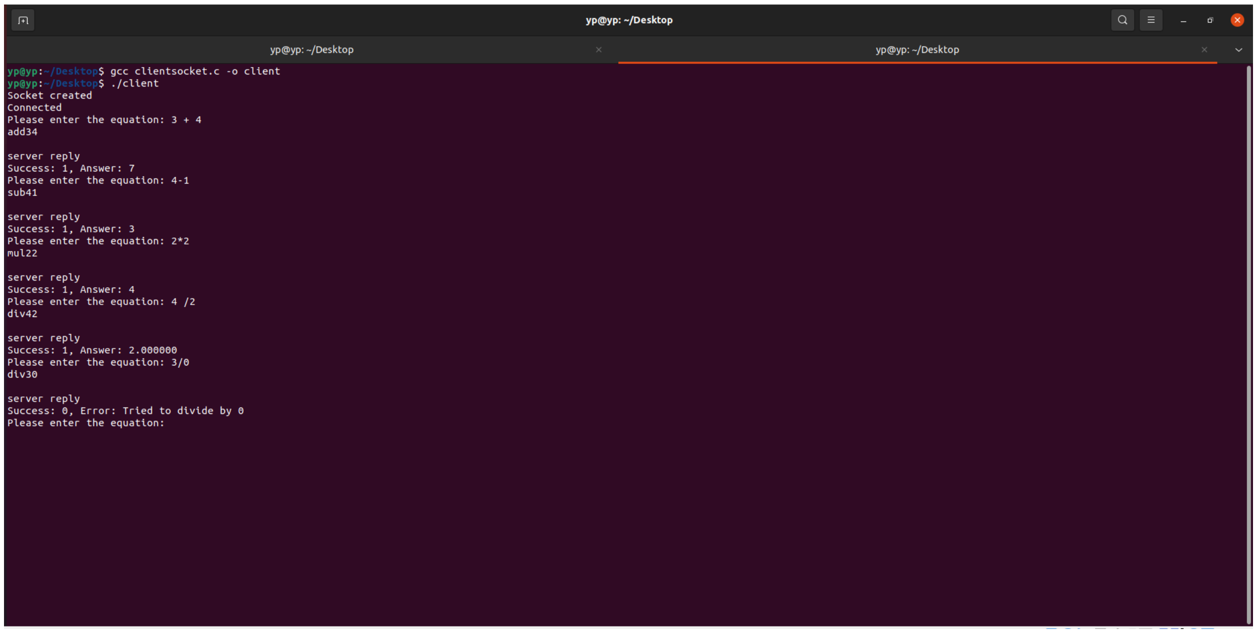


```c
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //divide
    //for when user enters multiplication equation
    //check if there is a / sign
    quotPtr = strchr(message, '/');
    //if there is a / sign
    if(quotPtr != NULL){
        //break the message var by the / sign, leaving only the numbers
        array[i] = strtok(message,"/");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"/");
        }
        //add "div" to the newstring variable
        strcat(newstring, "div");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //display the convert string
    puts(newstring);
    if (send(sock,newstring,strlen(newstring),0)< 0){
         printf("ERROR writing to socket");
        return 1;
        }

   if (recv(sock,server_reply,3000,0)<0){
        puts("recv failed");
        break;
        }
puts("server reply");
puts(server_reply);


```

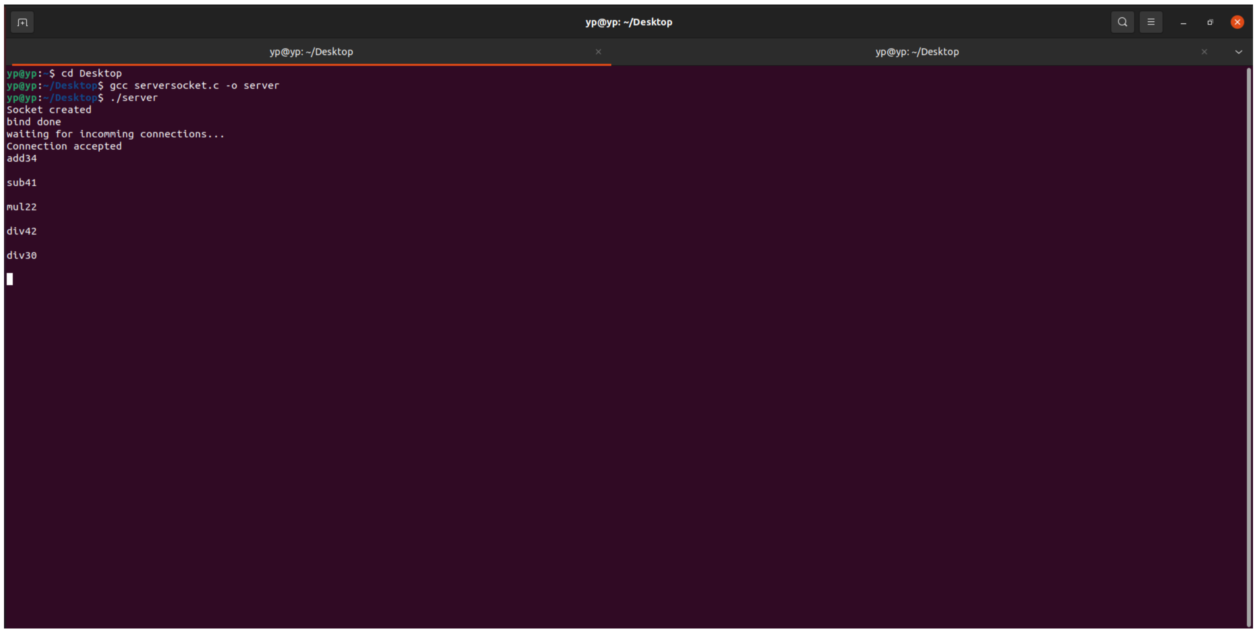


```c
puts("server reply");
puts(server_reply);


}
close(sock);
    return 0;
}


void remove_spaces(char* s) {
    char* d = s;
    do {
        while (*d == ' ') {
            ++d;
        }
    } while (*s++ = *d++);
}
```

Console



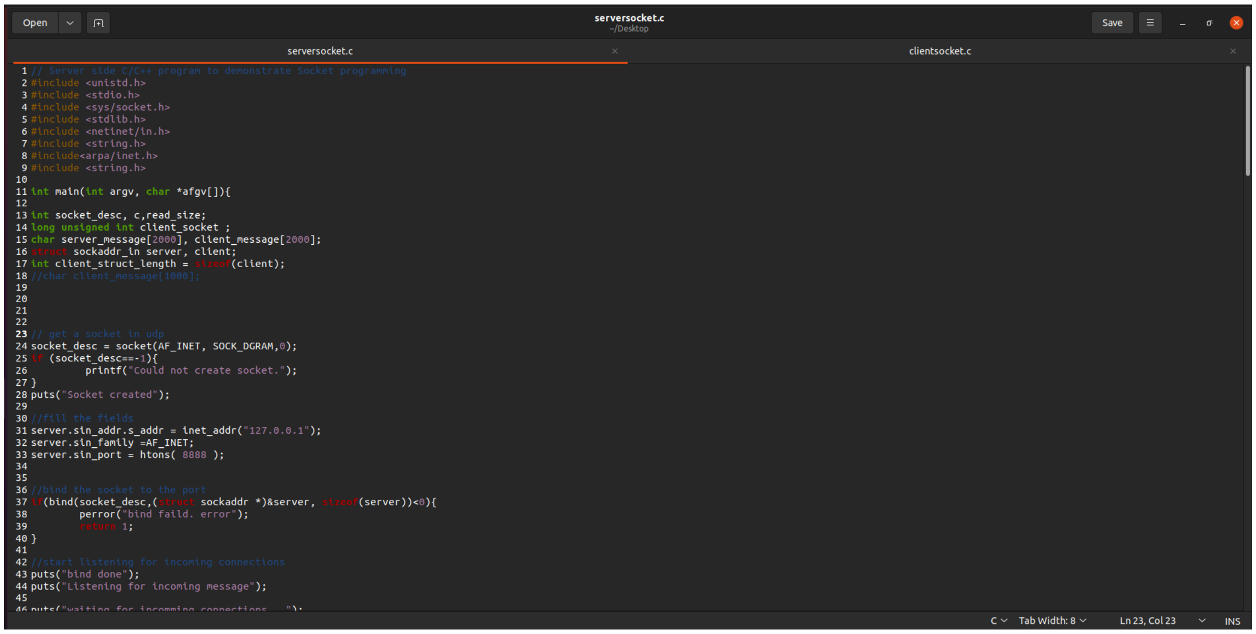
```
yp@yp:~/Desktop$ gcc clientsocket.c -o client
yp@yp:~/Desktop$ ./client
Socket created
Connected
Please enter the equation: 3 + 4
add34

server reply
Success: 1, Answer: 7
Please enter the equation: 4-1
sub41

server reply
Success: 1, Answer: 3
Please enter the equation: 2*2
mul22

server reply
Success: 1, Answer: 4
Please enter the equation: 4 /2
div42

server reply
Success: 1, Answer: 2.000000
Please enter the equation: 3/0
div30

server reply
Success: 0, Error: Tried to divide by 0
Please enter the equation:
```



```
yp@yp:~$ cd Desktop
yp@yp:~/Desktop$ gcc serversocket.c -o server
yp@yp:~/Desktop$ ./server
Socket created
bind done
waiting for incomming connections...
Connection accepted
add34

sub41

mul22

div42

div30

```

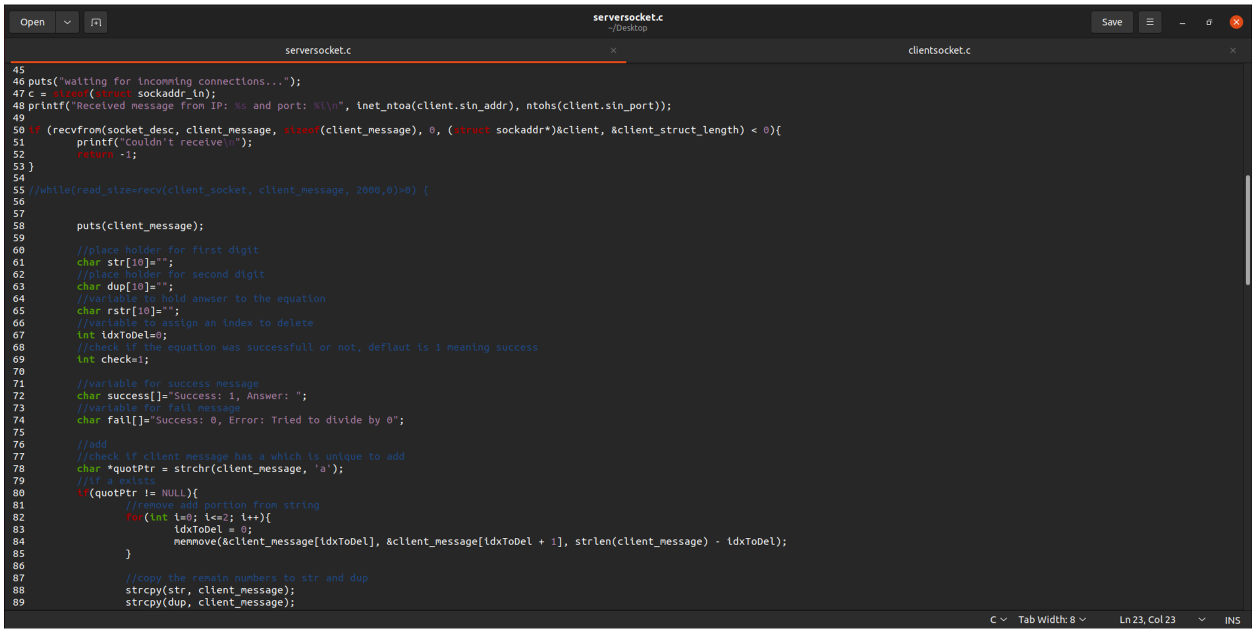
UDP:

Server

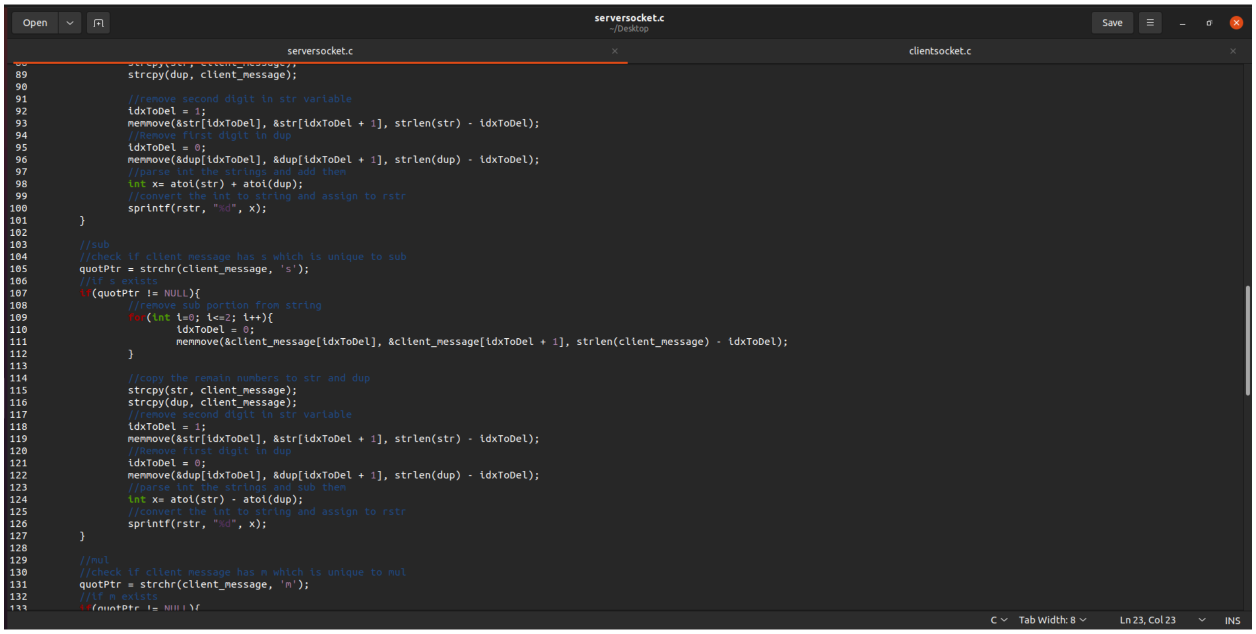


```c
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include<arpa/inet.h>
#include <string.h>

int main(int argv, char *afgv[]){

int socket_desc, c,read_size;
long unsigned int client_socket ;
char server_message[2000], client_message[2000];
struct sockaddr_in server, client;
int client_struct_length = sizeof(client);
//char client_message[1000];




// get a socket in udp
socket_desc = socket(AF_INET, SOCK_DGRAM,0);
if (socket_desc==-1){
        printf("Could not create socket.");
}
puts("Socket created");

//fill the fields
server.sin_addr.s_addr = inet_addr("127.0.0.1");
server.sin_family =AF_INET;
server.sin_port = htons( 8888 );


//bind the socket to the port
if(bind(socket_desc,(struct sockaddr *)&server, sizeof(server))<0){
        perror("bind faild. error");
        return 1;
}

//start listening for incoming connections
puts("bind done");
puts("Listening for incoming message");

puts("waiting for incomming connections...");
```

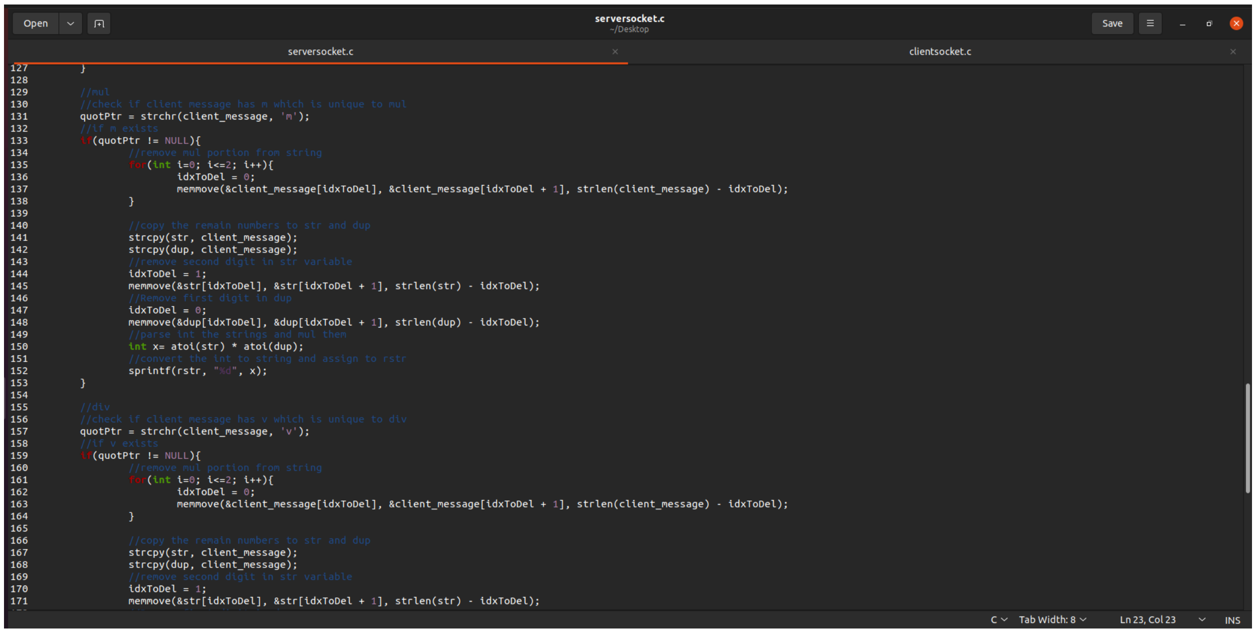
Yash Patel 100746810



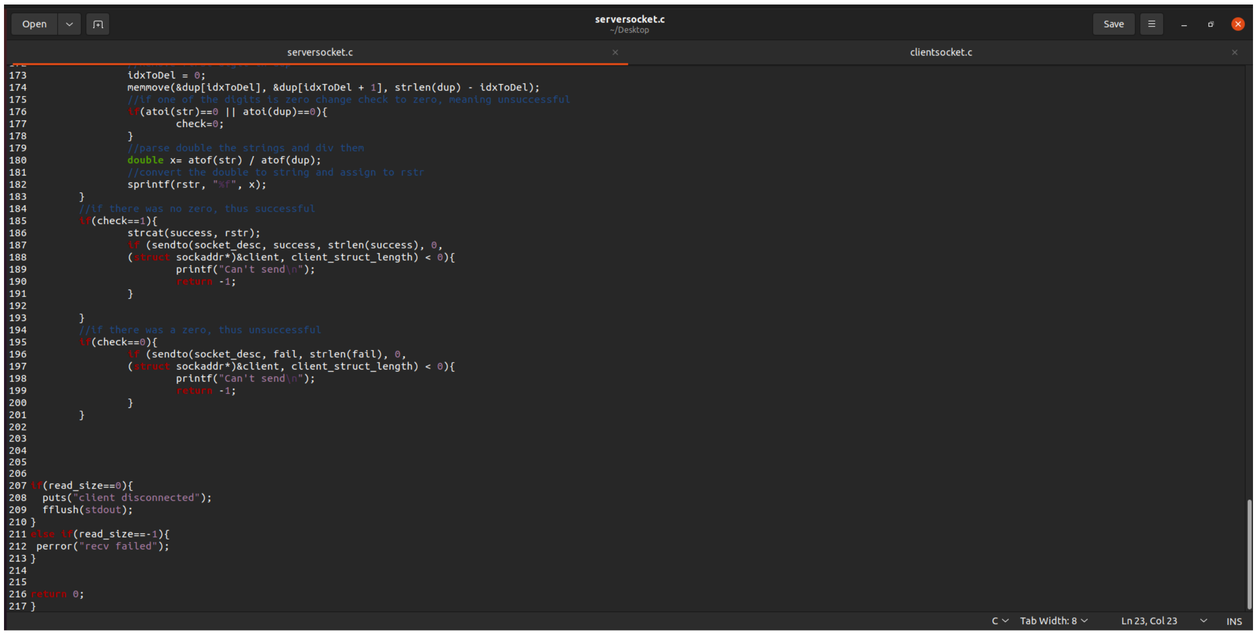
```c

puts("waiting for incomming connections...");
c = sizeof(struct sockaddr_in);
printf("Received message from IP: %s and port: %i\n", inet_ntoa(client.sin_addr), ntohs(client.sin_port));

if (recvfrom(socket_desc, client_message, sizeof(client_message), 0, (struct sockaddr*)&client, &client_struct_length) < 0){
        printf("Couldn't receive\n");
        return -1;
}

//while(read_size=recv(client_socket, client_message, 2000,0)>0) {


        puts(client_message);

        //place holder for first digit
        char str[10]="";
        //place holder for second digit
        char dup[10]="";
        //variable to hold anwser to the equation
        char rstr[10]="";
        //variable to assign an index to delete
        int idxToDel=0;
        //check if the equation was successfull or not, deflaut is 1 meaning success
        int check=1;

        //variable for success message
        char success[]="Success: 1, Answer: ";
        //variable for fail message
        char fail[]="Success: 0, Error: Tried to divide by 0";

        //add
        //check if client message has a which is unique to add
        char *quotPtr = strchr(client_message, 'a');
        //if a exists
        if(quotPtr != NULL){
                //remove add portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
```



```c
                strcpy(dup, client_message);

                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and add them
                int x= atoi(str) + atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //sub
        //check if client message has s which is unique to sub
        quotPtr = strchr(client_message, 's');
        //if s exists
        if(quotPtr != NULL){
                //remove sub portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and sub them
                int x= atoi(str) - atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //mul
        //check if client message has m which is unique to mul
        quotPtr = strchr(client_message, 'm');
        //if m exists
        if(quotPtr != NULL){
```



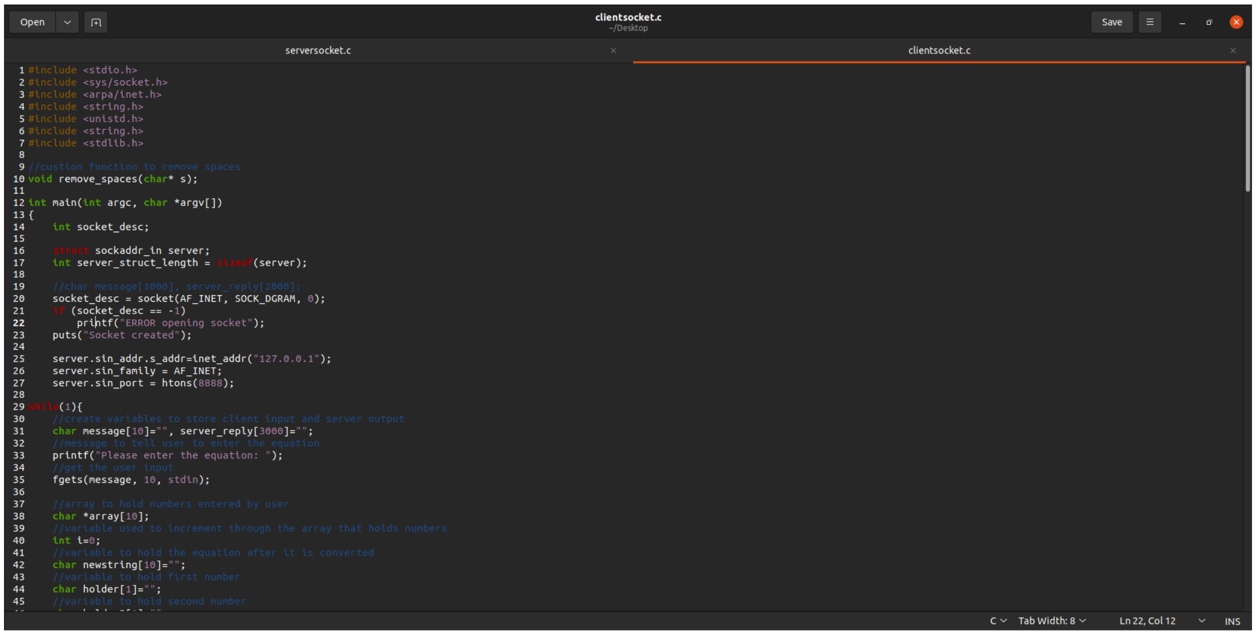
```c
        }

        //mul
        //check if client message has m which is unique to mul
        quotPtr = strchr(client_message, 'm');
        //if m exists
        if(quotPtr != NULL){
                //remove mul portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
                //Remove first digit in dup
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //parse int the strings and mul them
                int x= atoi(str) * atoi(dup);
                //convert the int to string and assign to rstr
                sprintf(rstr, "%d", x);
        }

        //div
        //check if client message has v which is unique to div
        quotPtr = strchr(client_message, 'v');
        //if v exists
        if(quotPtr != NULL){
                //remove mul portion from string
                for(int i=0; i<=2; i++){
                        idxToDel = 0;
                        memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
                }

                //copy the remain numbers to str and dup
                strcpy(str, client_message);
                strcpy(dup, client_message);
                //remove second digit in str variable
                idxToDel = 1;
                memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
```



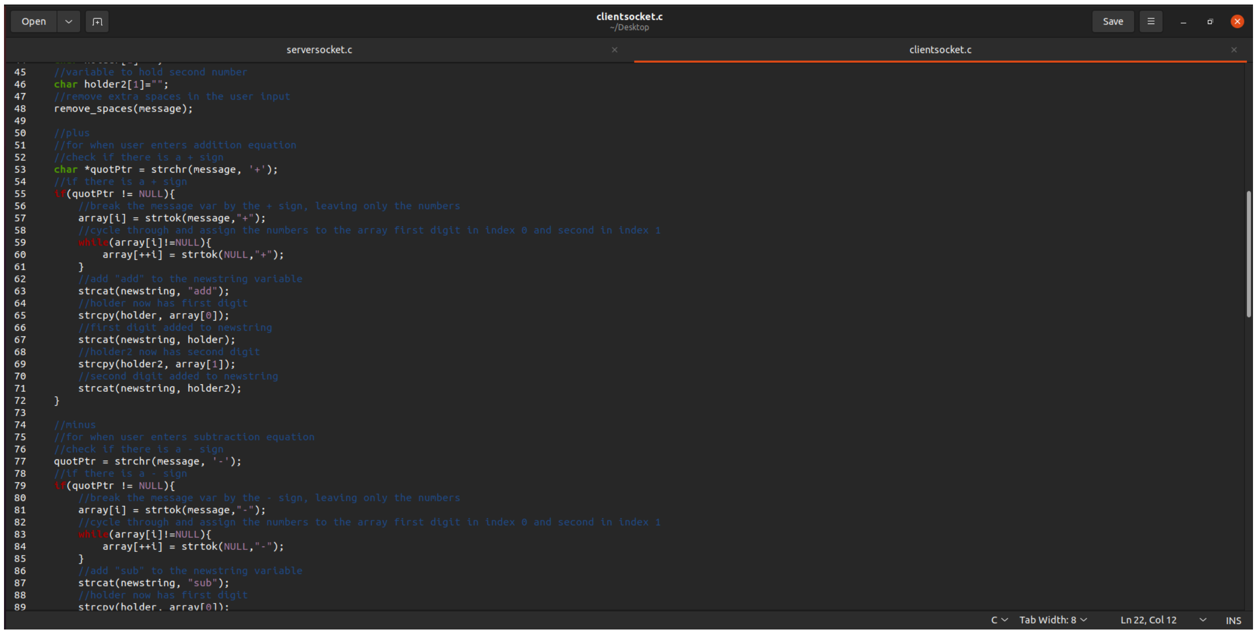
```c
                idxToDel = 0;
                memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
                //if one of the digits is zero change check to zero, meaning unsuccessful
                if(atoi(str)==0 || atoi(dup)==0){
                        check=0;
                }
                //parse double the strings and div them
                double x= atof(str) / atof(dup);
                //convert the double to string and assign to rstr
                sprintf(rstr, "%f", x);
        }
        //if there was no zero, thus successful
        if(check==1){
                strcat(success, rstr);
                if (sendto(socket_desc, success, strlen(success), 0,
                (struct sockaddr*)&client, client_struct_length) < 0){
                        printf("Can't send\n");
                        return -1;
                }

        }
        //if there was a zero, thus unsuccessful
        if(check==0){
                if (sendto(socket_desc, fail, strlen(fail), 0,
                (struct sockaddr*)&client, client_struct_length) < 0){
                        printf("Can't send\n");
                        return -1;
                }
        }





if(read_size==0){
  puts("client disconnected");
  fflush(stdout);
}
else if(read_size==-1){
 perror("recv failed");
}


return 0;
}
```

Client

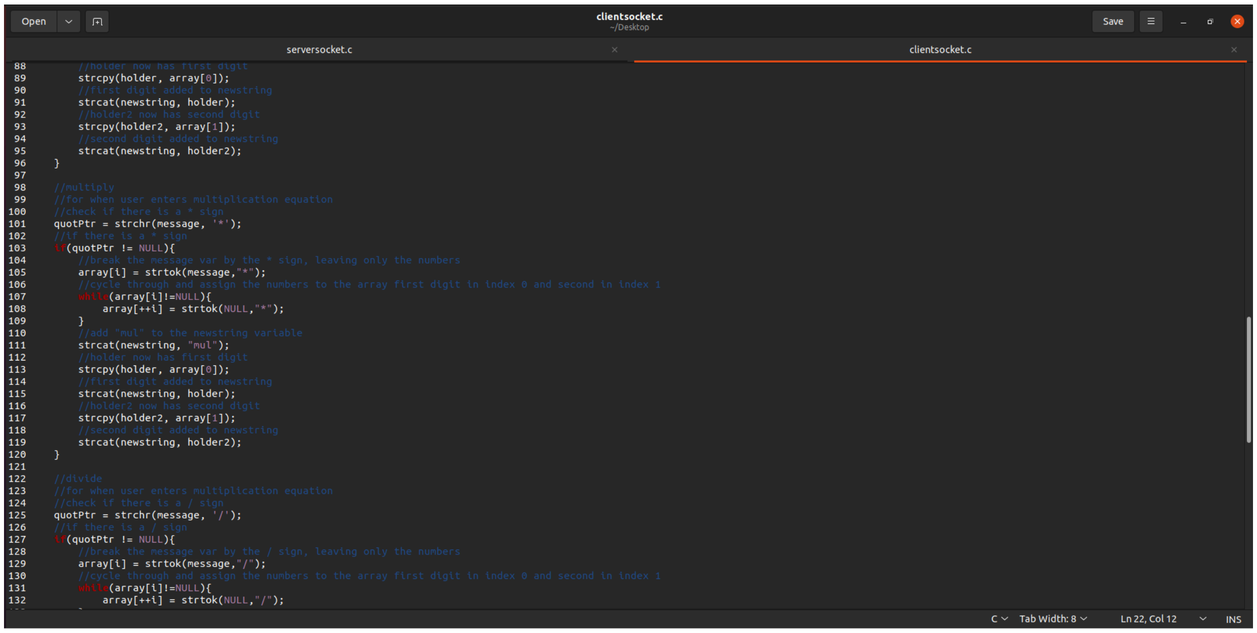
Yash Patel 100746810



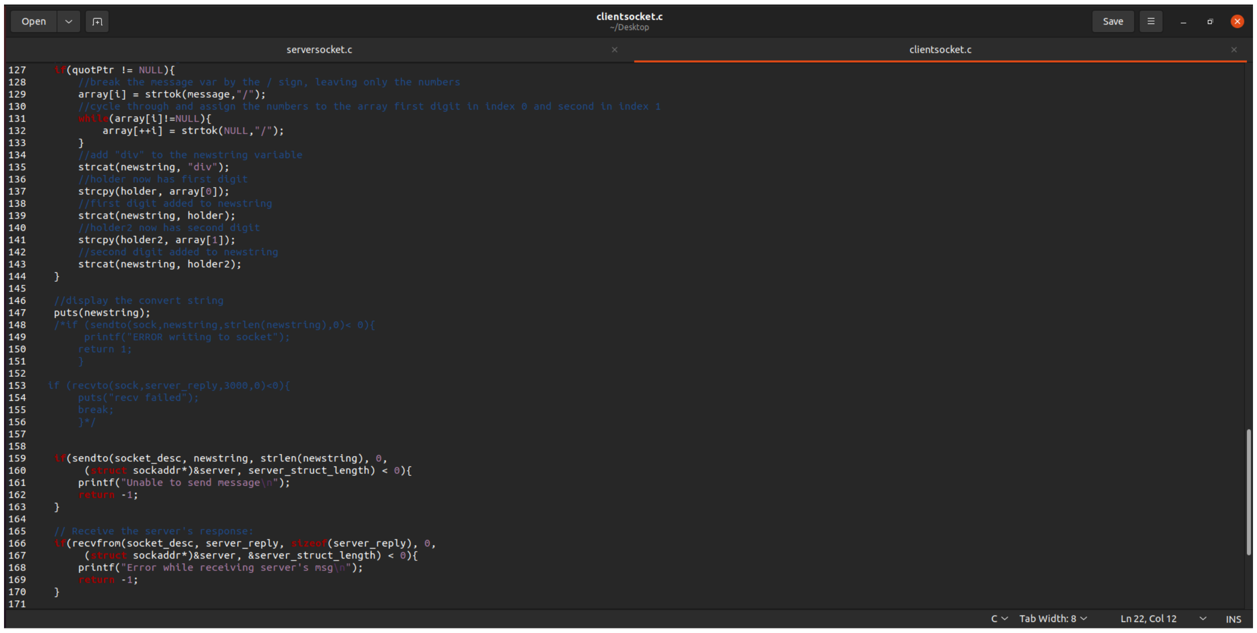
```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

//custion function to remove spaces
void remove_spaces(char* s);

int main(int argc, char *argv[])
{
    int socket_desc;

    struct sockaddr_in server;
    int server_struct_length = sizeof(server);

    //char message[1000], server_reply[2000];
    socket_desc = socket(AF_INET, SOCK_DGRAM, 0);
    if (socket_desc == -1)
        printf("ERROR opening socket");
    puts("Socket created");

    server.sin_addr.s_addr=inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons(8888);

while(1){
    //create variables to store client input and server output
    char message[10]="", server_reply[3000]="";
    //message to tell user to enter the equation
    printf("Please enter the equation: ");
    //get the user input
    fgets(message, 10, stdin);

    //array to hold numbers entered by user
    char *array[10];
    //variable used to increment through the array that holds numbers
    int i=0;
    //variable to hold the equation after it is converted
    char newstring[10]="";
    //variable to hold first number
    char holder[1]="";
    //variable to hold second number
```



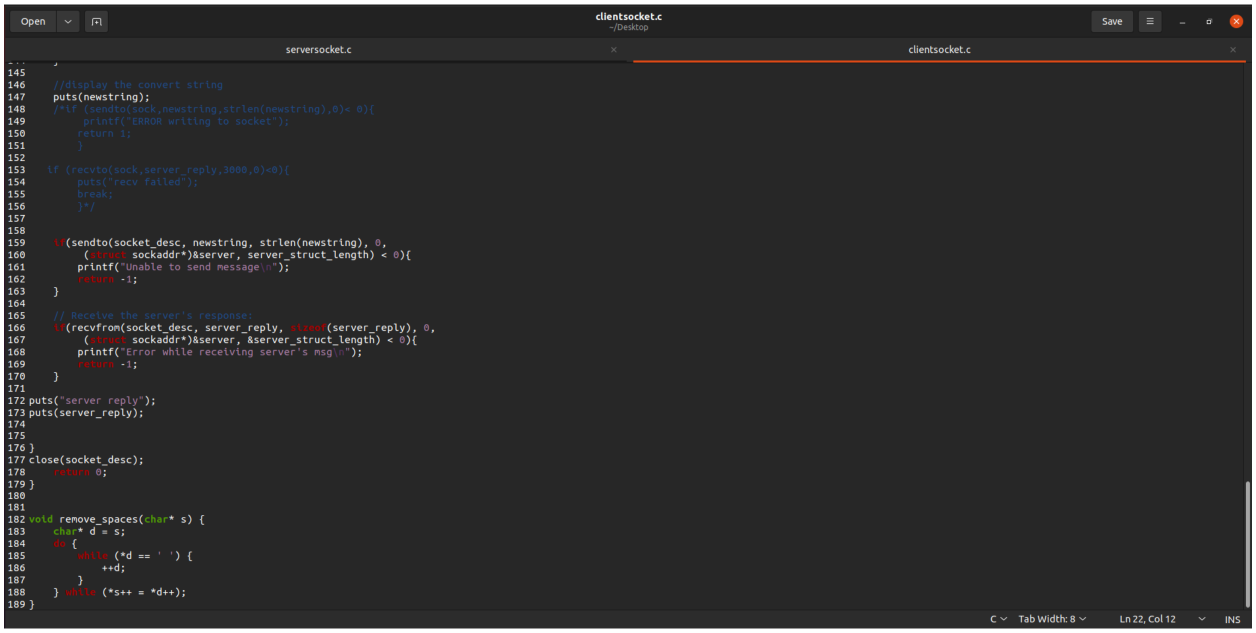
```c
    //variable to hold second number
    char holder2[1]="";
    //remove extra spaces in the user input
    remove_spaces(message);

    //plus
    //for when user enters addition equation
    //check if there is a + sign
    char *quotPtr = strchr(message, '+');
    //if there is a + sign
    if(quotPtr != NULL){
        //break the message var by the + sign, leaving only the numbers
        array[i] = strtok(message,"+");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"+");
        }
        //add "add" to the newstring variable
        strcat(newstring, "add");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //minus
    //for when user enters subtraction equation
    //check if there is a - sign
    quotPtr = strchr(message, '-');
    //if there is a - sign
    if(quotPtr != NULL){
        //break the message var by the - sign, leaving only the numbers
        array[i] = strtok(message,"-");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"-");
        }
        //add "sub" to the newstring variable
        strcat(newstring, "sub");
        //holder now has first digit
        strcpy(holder, array[0]);
```



```c
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //multiply
    //for when user enters multiplication equation
    //check if there is a * sign
    quotPtr = strchr(message, '*');
    //if there is a * sign
    if(quotPtr != NULL){
        //break the message var by the * sign, leaving only the numbers
        array[i] = strtok(message,"*");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"*");
        }
        //add "mul" to the newstring variable
        strcat(newstring, "mul");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //divide
    //for when user enters multiplication equation
    //check if there is a / sign
    quotPtr = strchr(message, '/');
    //if there is a / sign
    if(quotPtr != NULL){
        //break the message var by the / sign, leaving only the numbers
        array[i] = strtok(message,"/");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"/");
```

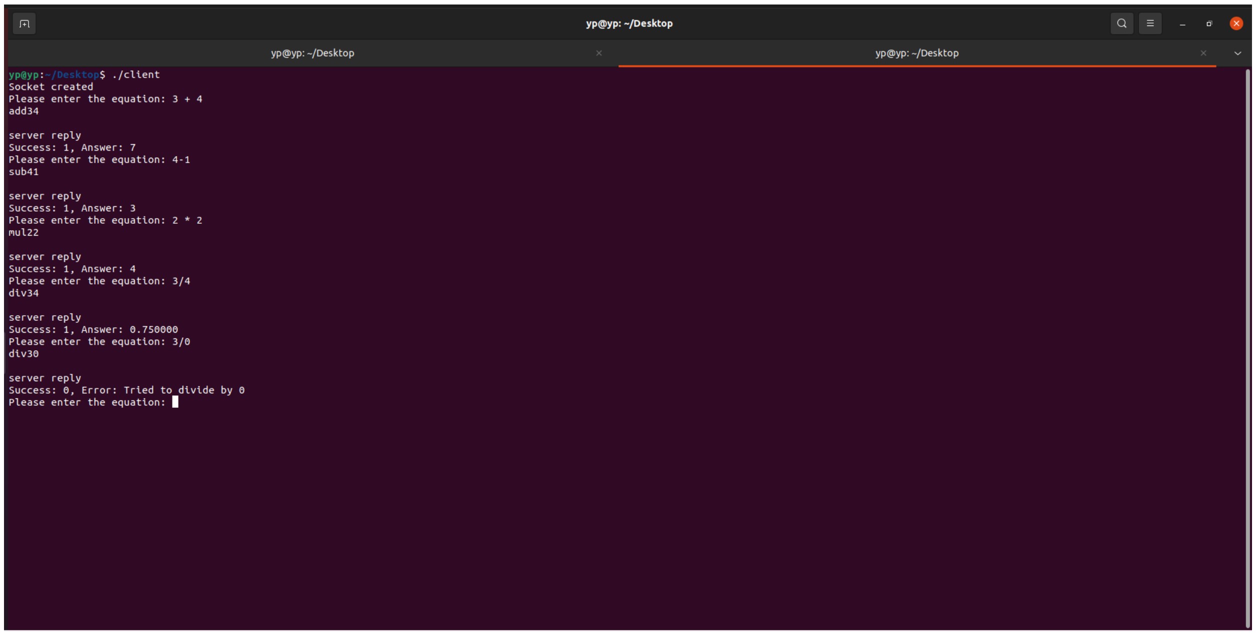


```c
    if(quotPtr != NULL){
        //break the message var by the / sign, leaving only the numbers
        array[i] = strtok(message,"/");
        //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
        while(array[i]!=NULL){
            array[++i] = strtok(NULL,"/");
        }
        //add "div" to the newstring variable
        strcat(newstring, "div");
        //holder now has first digit
        strcpy(holder, array[0]);
        //first digit added to newstring
        strcat(newstring, holder);
        //holder2 now has second digit
        strcpy(holder2, array[1]);
        //second digit added to newstring
        strcat(newstring, holder2);
    }

    //display the convert string
    puts(newstring);
    /*if (sendto(sock,newstring,strlen(newstring),0)< 0){
         printf("ERROR writing to socket");
        return 1;
        }

   if (recvto(sock,server_reply,3000,0)<0){
        puts("recv failed");
        break;
        }*/


    if(sendto(socket_desc, newstring, strlen(newstring), 0,
         (struct sockaddr*)&server, server_struct_length) < 0){
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server's response:
    if(recvfrom(socket_desc, server_reply, sizeof(server_reply), 0,
         (struct sockaddr*)&server, &server_struct_length) < 0){
        printf("Error while receiving server's msg\n");
        return -1;
    }

```



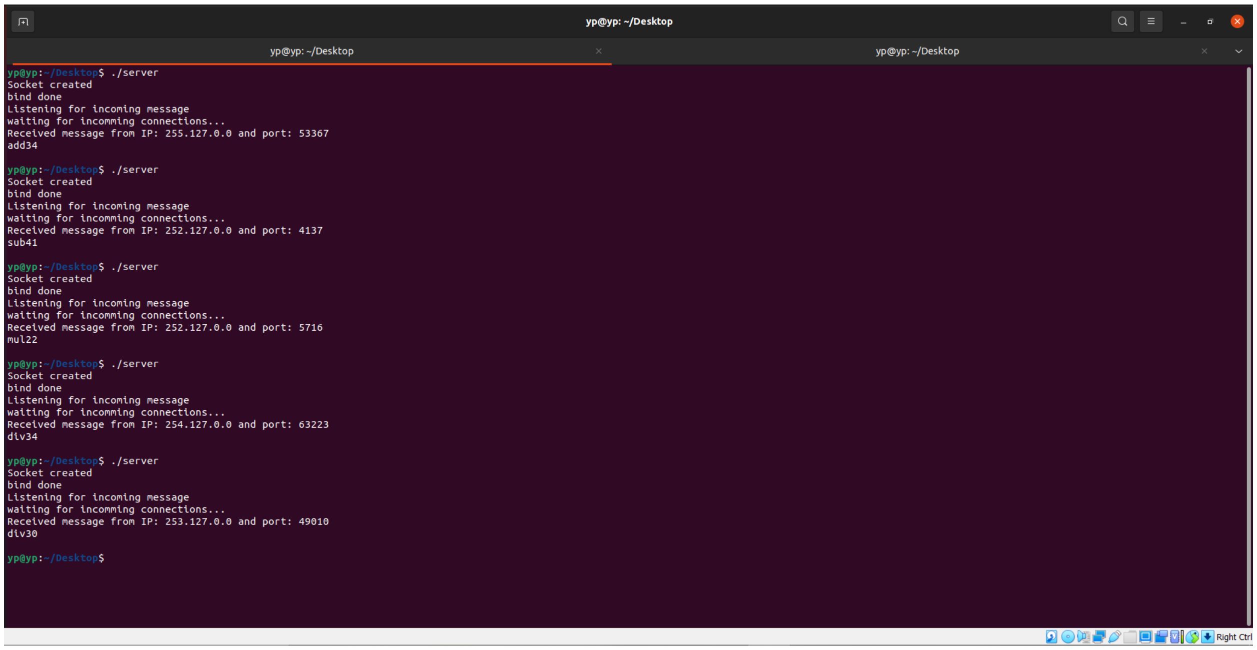
```c
    }

    //display the convert string
    puts(newstring);
    /*if (sendto(sock,newstring,strlen(newstring),0)< 0){
         printf("ERROR writing to socket");
        return 1;
        }

   if (recvto(sock,server_reply,3000,0)<0){
        puts("recv failed");
        break;
        }*/


    if(sendto(socket_desc, newstring, strlen(newstring), 0,
         (struct sockaddr*)&server, server_struct_length) < 0){
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server's response:
    if(recvfrom(socket_desc, server_reply, sizeof(server_reply), 0,
         (struct sockaddr*)&server, &server_struct_length) < 0){
        printf("Error while receiving server's msg\n");
        return -1;
    }

puts("server reply");
puts(server_reply);


}
close(socket_desc);
    return 0;
}


void remove_spaces(char* s) {
    char* d = s;
    do {
        while (*d == ' ') {
            ++d;
        }
    } while (*s++ = *d++);
}
```

Console



```
yp@yp:~/Desktop$ ./client
Socket created
Please enter the equation: 3 + 4
add34

server reply
Success: 1, Answer: 7
Please enter the equation: 4-1
sub41

server reply
Success: 1, Answer: 3
Please enter the equation: 2 * 2
mul22

server reply
Success: 1, Answer: 4
Please enter the equation: 3/4
div34

server reply
Success: 1, Answer: 0.750000
Please enter the equation: 3/0
div30

server reply
Success: 0, Error: Tried to divide by 0
Please enter the equation: 
```

Yash Patel 100746810



```
yp@yp:~/Desktop$ ./server
Socket created
bind done
Listening for incoming message
waiting for incomming connections...
Received message from IP: 255.127.0.0 and port: 53367
add34

yp@yp:~/Desktop$ ./server
Socket created
bind done
Listening for incoming message
waiting for incomming connections...
Received message from IP: 252.127.0.0 and port: 4137
sub41

yp@yp:~/Desktop$ ./server
Socket created
bind done
Listening for incoming message
waiting for incomming connections...
Received message from IP: 252.127.0.0 and port: 5716
mul22

yp@yp:~/Desktop$ ./server
Socket created
bind done
Listening for incoming message
waiting for incomming connections...
Received message from IP: 254.127.0.0 and port: 63223
div34

yp@yp:~/Desktop$ ./server
Socket created
bind done
Listening for incoming message
waiting for incomming connections...
Received message from IP: 253.127.0.0 and port: 49010
div30

yp@yp:~/Desktop$
```