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,n = 0,num=0;

char recvBuff[1024];

char sendBuff[1025];

struct sockaddr\_in serv\_addr;

memset(recvBuff, '0' ,sizeof(recvBuff));

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);

serv\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if(connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0){

printf("\n Error : Connect Failed \n");

return 1;

}

printf("Welcome to our Mathematics server\n");

num = recv(sockfd, recvBuff, sizeof(recvBuff),0);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

printf("%s\n",recvBuff);

recvBuff[0] = '\0';

printf("\nEnter the numbers and operation : ");

fgets(sendBuff,1025,stdin);

if ((send(sockfd,sendBuff,strlen(sendBuff),0))== -1){fprintf(stderr, "Failure Sending Message\n");

close(sockfd);

}

num = recv(sockfd, recvBuff, sizeof(recvBuff),0);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

printf("%s",recvBuff);

close(sockfd);

return 0;

}

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>

#include <time.h>

struct customer {

char operation[40];

char cli\_addr[16];

int cli\_port;

char time[80];

struct customer \*next;

};

struct customer \*Create\_Customer(){

struct customer \*x =(struct customer \*)malloc(sizeof(struct customer));

x->next=NULL;

return x;

}

int main(){

time\_t now;

struct tm \*ts;

char buf[80];

int listenfd = 0,connfd = 0,n=0,num=0;

struct sockaddr\_in serv\_addr;

struct sockaddr\_in client\_addr;

char sendBuff[1025];

char recvBuff[1024];

int numrv;

char new1[50],num2[10],num3[10];

int newn,t,h,j,i,n1,n2;

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;

}

printf("\nAvailable operations : ");

printf("\nAddition(+)");

printf("\nSubstraction(-)");

printf("\nMultiplication(\*)");

printf("\nDivision(/)");

struct customer \*\*Customers,\*head,\*temp;

Customers = &head;

int customer\_count = 0;

while(1){

socklen\_t addr\_size = sizeof(client\_addr);

connfd = accept(listenfd, (struct sockaddr\*)&client\_addr, &addr\_size);

struct sockaddr\_in\* pV4Addr = (struct sockaddr\_in\*)&client\_addr;

struct in\_addr ipAddr = pV4Addr->sin\_addr;

char str[INET\_ADDRSTRLEN];

inet\_ntop( AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN );

sprintf(sendBuff,"Number of Unique Customers so far: %d",customer\_count);

if ((send(connfd,sendBuff,strlen(sendBuff),0))== -1){

fprintf(stderr, "Failure Sending Message\n");

close(connfd);

break;

}

num = recv(connfd, recvBuff, sizeof(recvBuff),0);now = time(NULL);//time-stamping

ts = localtime(&now);

strftime(buf, sizeof(buf), "%a %Y-%m-%d %H:%M:%S %Z", ts);

if ( num <= 0 ){

printf("Either Connection Closed or Error\n");

}

recvBuff[num] = '\0';

i=0;

if(!(recvBuff[i]=='+'||recvBuff[i]=='-'||recvBuff[i]=='\*'||recvBuff[i]=='/'))

{

printf("\nWrong input");

return 0;

}

i=1;

j=0;

while(recvBuff[i]!=' ')

num2[j++]=recvBuff[i++];

num2[j]='\0';

j=0;

while(recvBuff[i]!='\0')

num3[j++]=recvBuff[i++];

num3[j]='\0';

printf("\n Num1, Num2 :%s, %s",num2,num3);

n1=atoi(num2);

n2=atoi(num3);

if(recvBuff[0]=='+'){

t=n1+n2;

\*Customers = Create\_Customer();

strcpy((\*Customers)->operation,"Addition");

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"The result is : %d",t);

}

if(recvBuff[0]=='-')

{

t=n1-n2;

\*Customers = Create\_Customer();

strcpy((\*Customers)->operation,"Substraction");

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"The result is : %d",t);

}

if(recvBuff[0]=='\*'){

t=n1\*n2;

\*Customers = Create\_Customer();

strcpy((\*Customers)->operation,"Multiplication");

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"The result is : %d",t);

}

if(recvBuff[0]=='/'){

t=n1/n2;

\*Customers = Create\_Customer();

strcpy((\*Customers)->operation,"Division");

strcpy((\*Customers)->cli\_addr,str);

(\*Customers)->cli\_port = client\_addr.sin\_port;

strcpy((\*Customers)->time,buf);

Customers=&((\*Customers)->next);

customer\_count++;

sprintf(sendBuff,"The result is : %d",t);

}

h=t;

struct customer \*t;

t=head;

int x=0;

while(t){

x++;

printf("\n===============%d===========",x);

printf("\nCustomer IP: %s",t->cli\_addr);

printf("\nCustomer port: %d",t->cli\_port);

printf("\nOperation done: %s",t->operation);

printf("\nResult: %d",h);

printf("\nTime-Stamp: %s",t->time);

printf("\n\n");

t=t->next;

}

if ((send(connfd,sendBuff,strlen(sendBuff),0))== -1){

fprintf(stderr, "Failure Sending Message\n");

close(connfd);

break;

}

close(connfd);

sleep(1);

}

return 0;

}

OUTPUT:

