Njungle Nges Brian

FE14A169

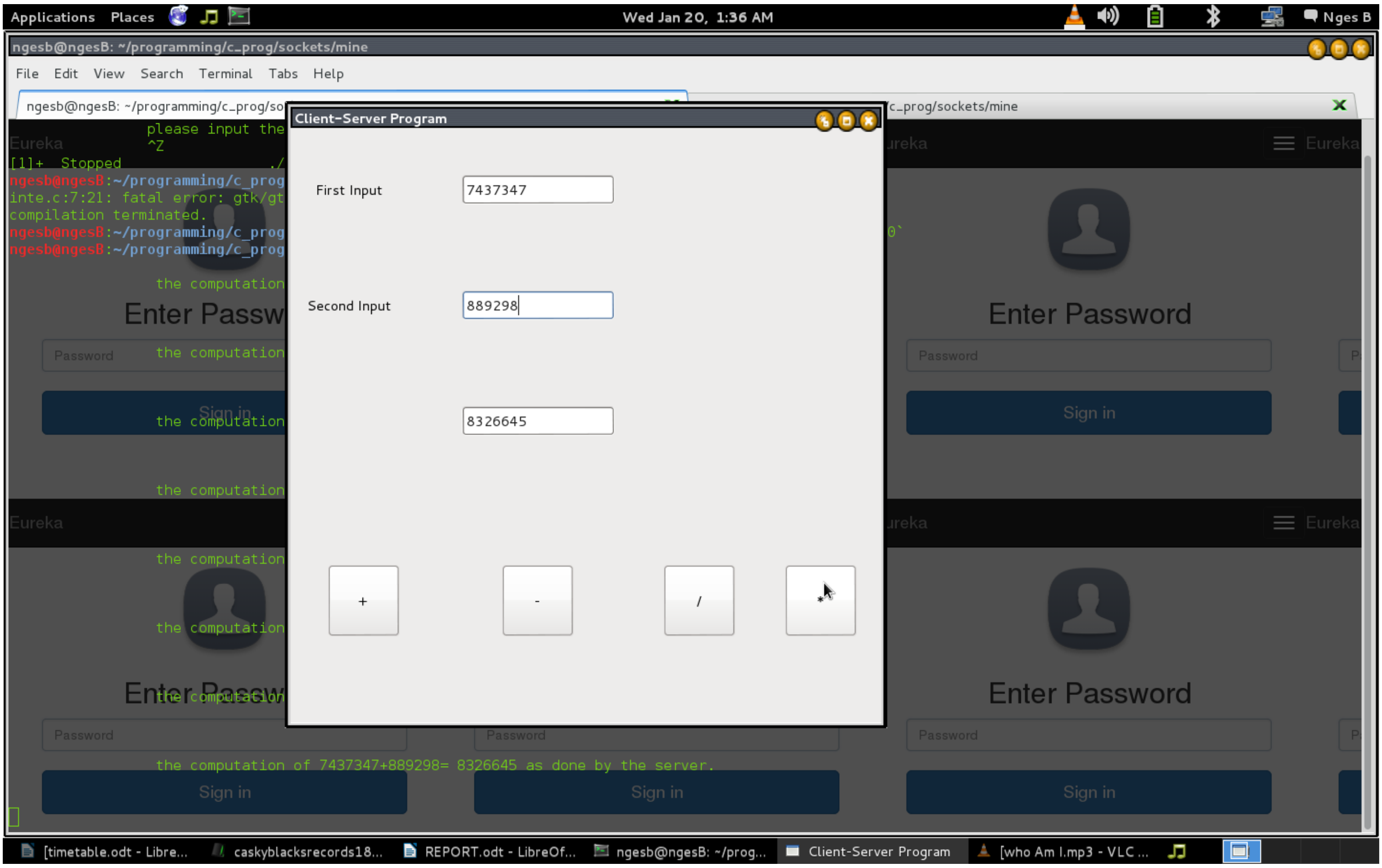
CEF305 ASSIGEMENT
TITTLE:
A CLIENT SERVER PROGRAM WITH A GUI DONE IN SOCKET PROGRAMING AND GTK.

PROPERTIES OF MECHINE

Operating system:Kali Linux

Labtop properties :Dell Latitude 6410,Hard drive,320Gb, 5processors,RAM=4Gb

Pictures of the interface



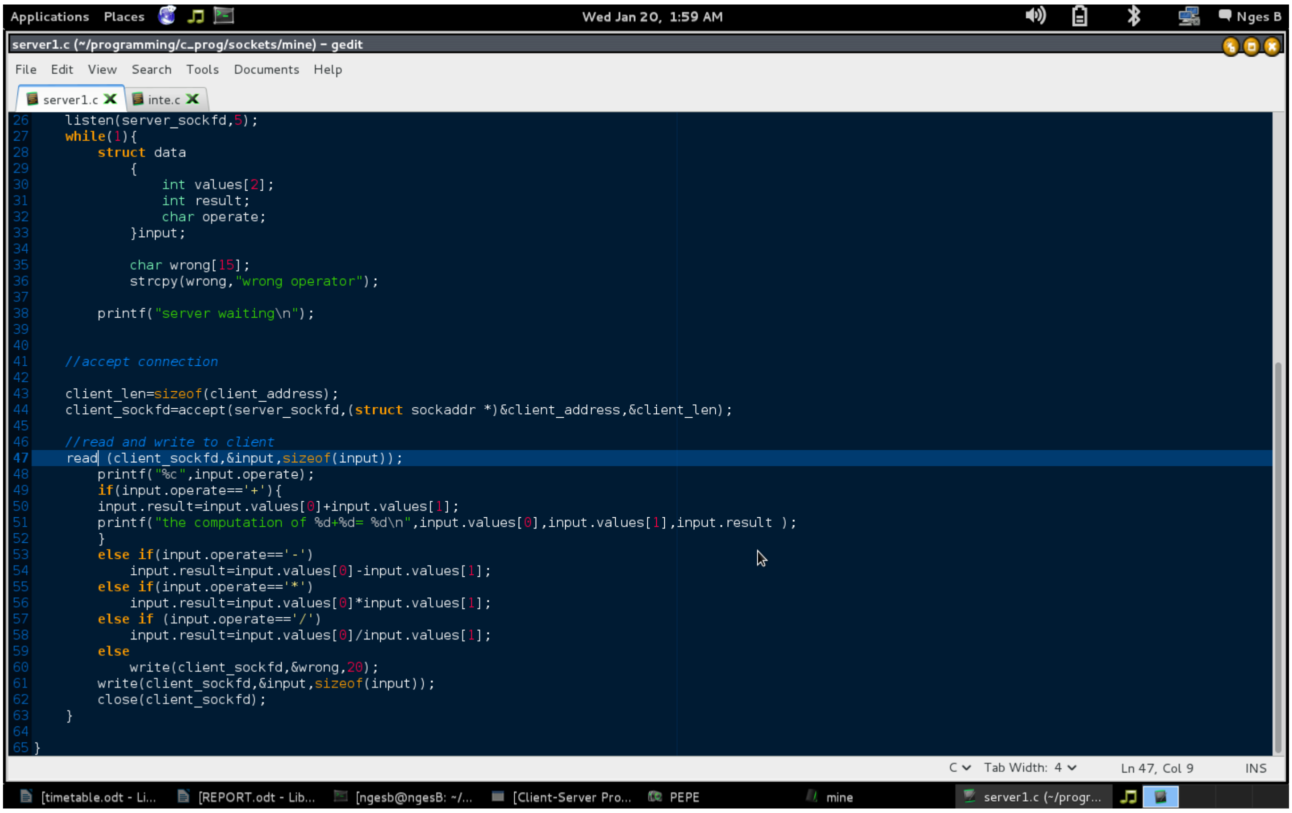
```
              please input the
              ^Z
[1]+  Stopped                  ./
ngesb@ngesB:~/programming/c_prog
inte.c:7:21: fatal error: gtk/gt
compilation terminated.
ngesb@ngesB:~/programming/c_prog
ngesb@ngesB:~/programming/c_prog

                the computation

                the computation

                the computation

                the computation

                the computation

                the computation

                the computation

                the computation of 7437347+889298= 8326645 as done by the server.
```

First Input: 7437347

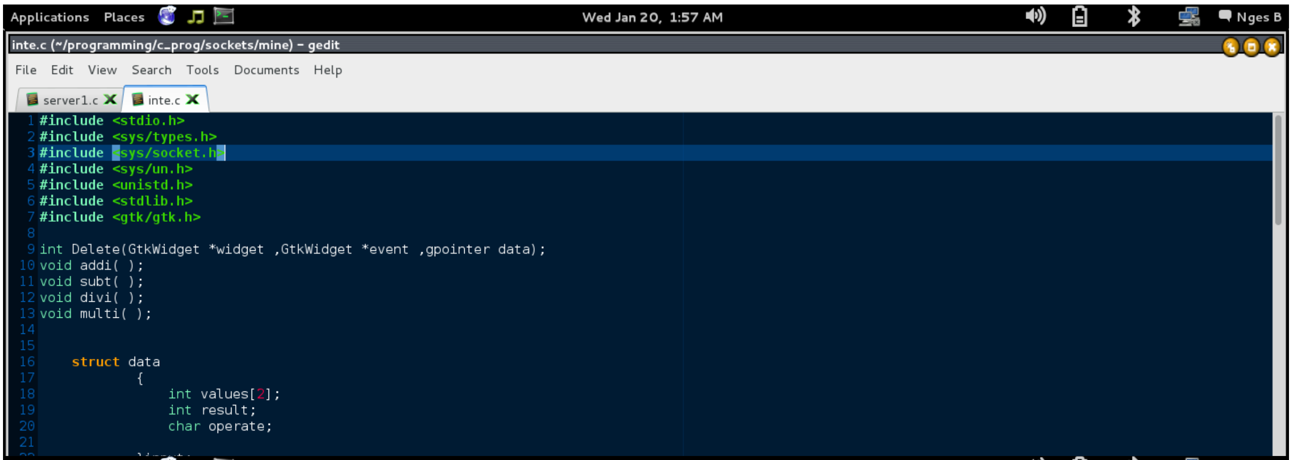
Second Input: 889298

8326645

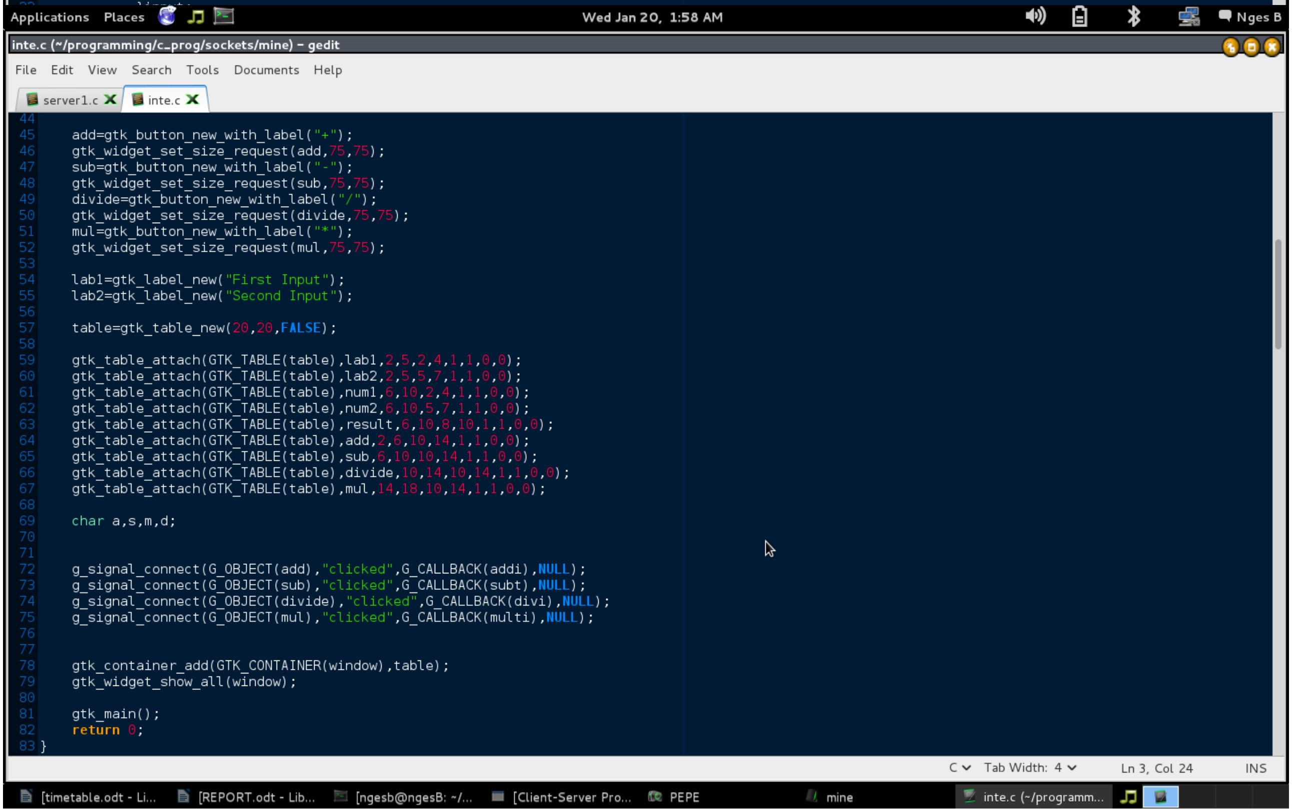
Pictures of Codes



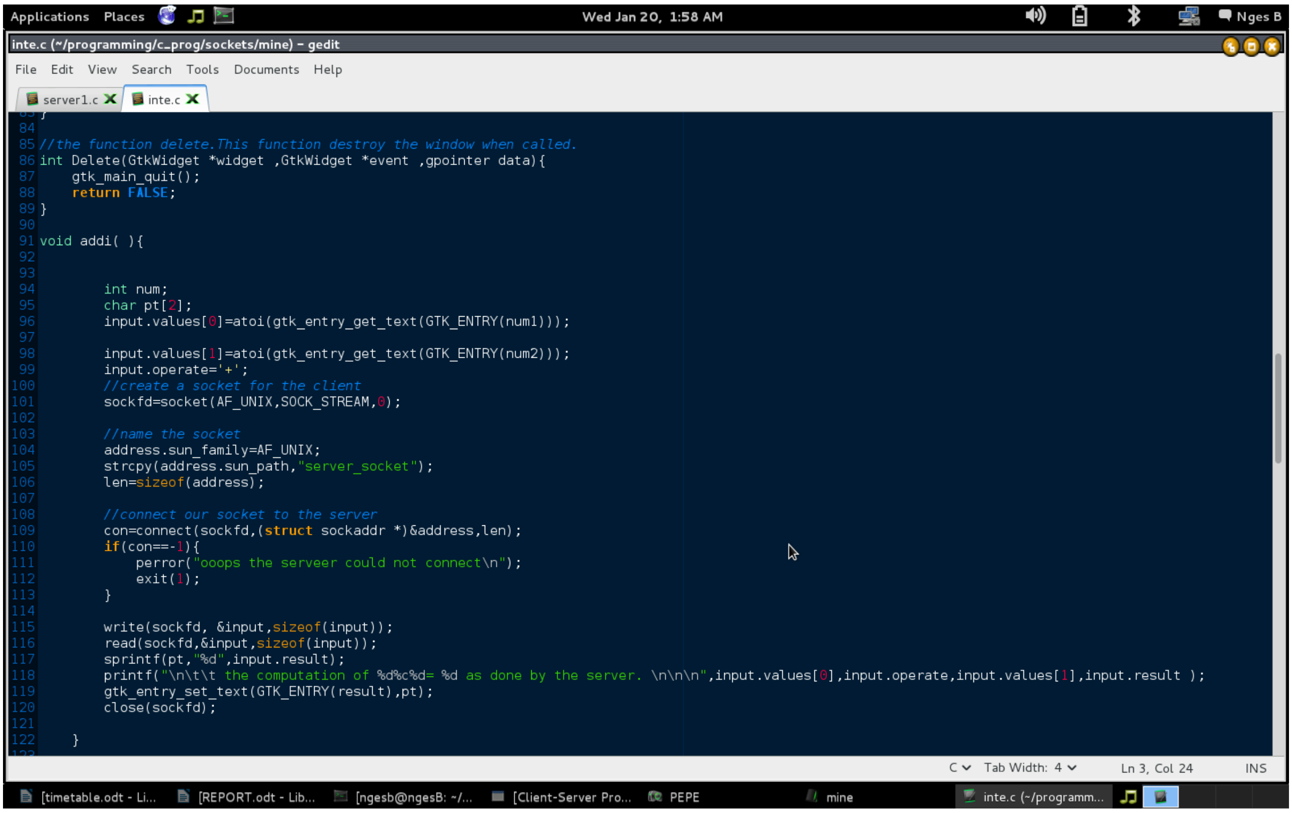
```
    listen(server_sockfd,5);
    while(1){
        struct data
            {
                int values[2];
                int result;
                char operate;
            }input;

            char wrong[15];
            strcpy(wrong,"wrong operator");

        printf("server waiting\n");


    //accept connection

    client_len=sizeof(client_address);
    client_sockfd=accept(server_sockfd,(struct sockaddr *)&client_address,&client_len);

    //read and write to client
    read (client_sockfd,&input,sizeof(input));
        printf("%c",input.operate);
        if(input.operate=='+'){
        input.result=input.values[0]+input.values[1];
        printf("the computation of %d+%d= %d\n",input.values[0],input.values[1],input.result );
        }
        else if(input.operate=='-')
            input.result=input.values[0]-input.values[1];
        else if(input.operate=='*')
            input.result=input.values[0]*input.values[1];
        else if (input.operate=='/')
            input.result=input.values[0]/input.values[1];
        else
            write(client_sockfd,&wrong,20);
        write(client_sockfd,&input,sizeof(input));
        close(client_sockfd);
    }

}
```



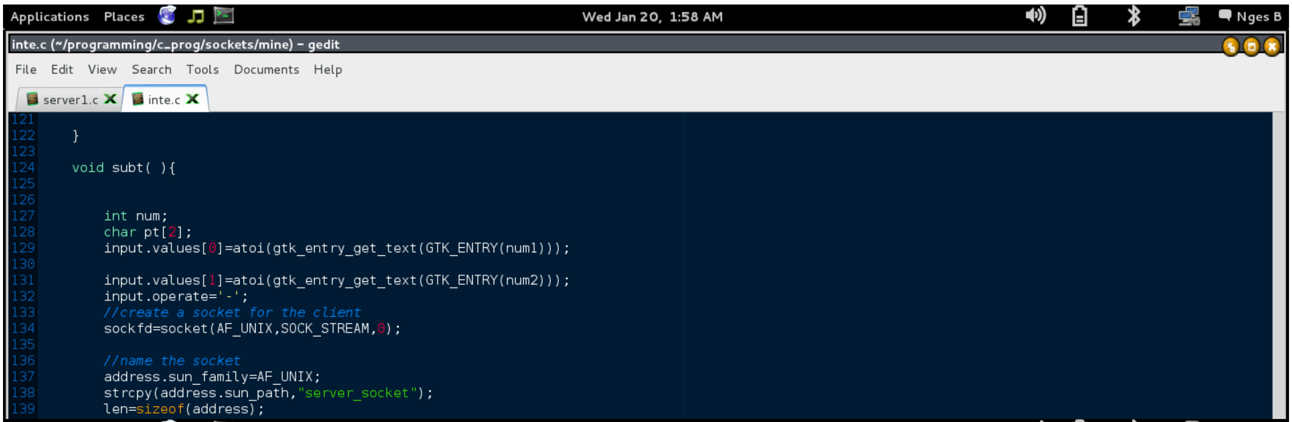
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
#include <gtk/gtk.h>

int Delete(GtkWidget *widget ,GtkWidget *event ,gpointer data);
void addi( );
void subt( );
void divi( );
void multi( );


    struct data
            {
                int values[2];
                int result;
                char operate;
```



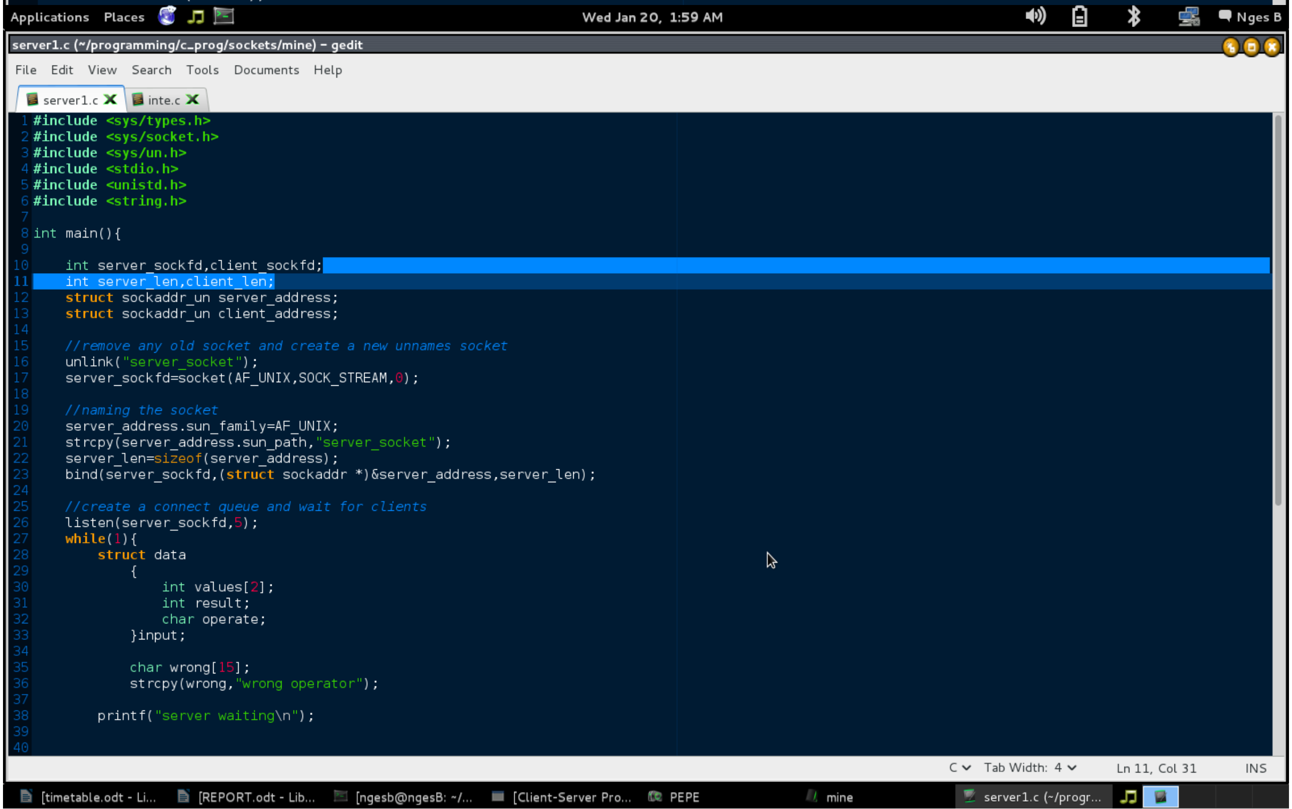
```

    add=gtk_button_new_with_label("+");
    gtk_widget_set_size_request(add,75,75);
    sub=gtk_button_new_with_label("-");
    gtk_widget_set_size_request(sub,75,75);
    divide=gtk_button_new_with_label("/");
    gtk_widget_set_size_request(divide,75,75);
    mul=gtk_button_new_with_label("*");
    gtk_widget_set_size_request(mul,75,75);

    lab1=gtk_label_new("First Input");
    lab2=gtk_label_new("Second Input");

    table=gtk_table_new(20,20,FALSE);

    gtk_table_attach(GTK_TABLE(table),lab1,2,5,2,4,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),lab2,2,5,5,7,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),num1,6,10,2,4,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),num2,6,10,5,7,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),result,6,10,8,10,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),add,2,6,10,14,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),sub,6,10,10,14,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),divide,10,14,10,14,1,1,0,0);
    gtk_table_attach(GTK_TABLE(table),mul,14,18,10,14,1,1,0,0);

    char a,s,m,d;


    g_signal_connect(G_OBJECT(add),"clicked",G_CALLBACK(addi),NULL);
    g_signal_connect(G_OBJECT(sub),"clicked",G_CALLBACK(subt),NULL);
    g_signal_connect(G_OBJECT(divide),"clicked",G_CALLBACK(divi),NULL);
    g_signal_connect(G_OBJECT(mul),"clicked",G_CALLBACK(multi),NULL);


    gtk_container_add(GTK_CONTAINER(window),table);
    gtk_widget_show_all(window);

    gtk_main();
    return 0;
}
```



```
//the function delete.This function destroy the window when called.
int Delete(GtkWidget *widget ,GtkWidget *event ,gpointer data){
    gtk_main_quit();
    return FALSE;
}

void addi( ){


        int num;
        char pt[2];
        input.values[0]=atoi(gtk_entry_get_text(GTK_ENTRY(num1)));

        input.values[1]=atoi(gtk_entry_get_text(GTK_ENTRY(num2)));
        input.operate='+';
        //create a socket for the client
        sockfd=socket(AF_UNIX,SOCK_STREAM,0);

        //name the socket
        address.sun_family=AF_UNIX;
        strcpy(address.sun_path,"server_socket");
        len=sizeof(address);

        //connect our socket to the server
        con=connect(sockfd,(struct sockaddr *)&address,len);
        if(con==-1){
            perror("ooops the serveer could not connect\n");
            exit(1);
        }

        write(sockfd, &input,sizeof(input));
        read(sockfd,&input,sizeof(input));
        sprintf(pt,"%d",input.result);
        printf("\n\t\t the computation of %d%c%d= %d as done by the server. \n\n\n",input.values[0],input.operate,input.values[1],input.result );
        gtk_entry_set_text(GTK_ENTRY(result),pt);
        close(sockfd);

    }
```



```

    }

    void subt( ){


        int num;
        char pt[2];
        input.values[0]=atoi(gtk_entry_get_text(GTK_ENTRY(num1)));

        input.values[1]=atoi(gtk_entry_get_text(GTK_ENTRY(num2)));
        input.operate='-';
        //create a socket for the client
        sockfd=socket(AF_UNIX,SOCK_STREAM,0);

        //name the socket
        address.sun_family=AF_UNIX;
        strcpy(address.sun_path,"server_socket");
        len=sizeof(address);
```



```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(){

    int server_sockfd,client_sockfd;
    int server_len,client_len;
    struct sockaddr_un server_address;
    struct sockaddr_un client_address;

    //remove any old socket and create a new unnames socket
    unlink("server_socket");
    server_sockfd=socket(AF_UNIX,SOCK_STREAM,0);

    //naming the socket
    server_address.sun_family=AF_UNIX;
    strcpy(server_address.sun_path,"server_socket");
    server_len=sizeof(server_address);
    bind(server_sockfd,(struct sockaddr *)&server_address,server_len);

    //create a connect queue and wait for clients
    listen(server_sockfd,5);
    while(1){
        struct data
            {
                int values[2];
                int result;
                char operate;
            }input;

            char wrong[15];
            strcpy(wrong,"wrong operator");

        printf("server waiting\n");


```