# CSE 489/589
# Programming Assignment 1 Report
# Group No.: 60
# Text Chat Application

**Notes: (IMPORTANT)**

- ➔ One of your group members select <File> - <Make a copy> to make a copy of this report for your group, and share that Google Doc copy with your teammates so that they can also edit it.
- ➔ Report your work in each section. Describe the method you used, the obstacles you met, how you solved them, and the results. You can take screenshots at key points. There are NO hard requirements for your description.
- ➔ For a certain command/event, if you successfully implemented it, **take a screenshot of the result from the grader (required).** You will get full points if it can pass the corresponding test case of the automated grader.
- ➔ For a certain command/event, if you tried but failed to implement it, properly describe your work. We will partially grade it based on the work you did.
- ➔ Do **NOT** claim anything you didn't implement. If you didn't try on a certain command or event, leave that section blank. We will randomly check your code, and if it does not match the work you claimed, you and your group won't get any partial grade score for this WHOLE assignment.
- ➔ There will be 10 bonus points for this report. Grading will be based on the organization, presentation, and layout of your report.
- ➔ After you finish, export this report as a PDF file and submit it on the UBLearns. For each group, only one member needs to make the submission.

## 1 - Group and Contributions

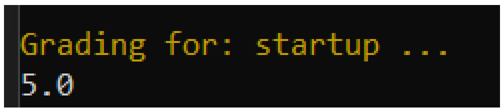
- Name of member 1:
  - avadhari
  - Client Related Commands and Code
- Name of member 2:
  - akumar59:
  - Server Related Commands and Code

## 2 - SHELL Functionality

### [5.0] Application Startup

This test will check whether the application is starting or not properly when u issue the below commands

**For server:**
./assignment1 s “portno”
**For client:**
./assignment1 c “portno”

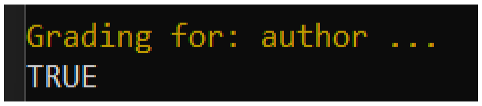


```
Grading for: startup ...
5.0
```

## 3 - Command for Server and Client

### [0.0] AUTHOR

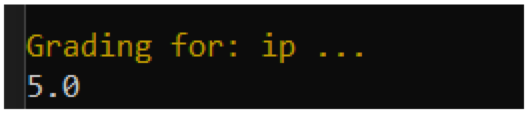
We print the author name when client or server types “AUTHOR”, this test is not numerically graded and we are receiving ‘True’ as a Grade.



```
Grading for: author ...
TRUE
```

### [5.0] IP

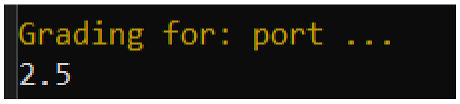
We store and keep the ip address in a structure and display it when someone types “IP”.



```
Grading for: ip ...
5.0
```

### [2.5] PORT

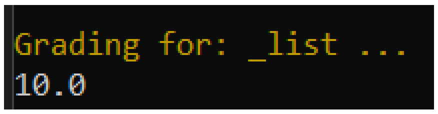
We store and keep the port number in a structure and display it when someone types “PORT”.



```
Grading for: port ...
2.5
```

### [10.0] LIST

When someone types “LIST” whether server or client it will iterate and list all the logged in clients to the server. The server Maintains a list of Logged in clients.

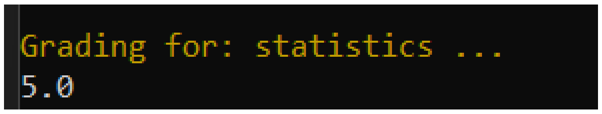


```
Grading for: _list ...
10.0
```

## 4 - Command/Event for Server

### [5.0] STATISTICS

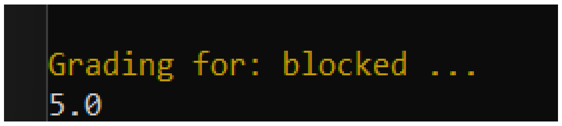
For  Statistics it will iterate over the list of logged in clients who have not yet excited.
It will iterate over all the clients whether logged in or logged-out and will tell how many messages a client has received or sent.



```
Grading for: statistics ...
5.0
```

### [7.0] BLOCKED <client-ip> + Exception Handling

We keep a structure at infm in a global class which contains block_list  and client and server extends global class. So each client has a list of blocked clients. To run this code you run “BLOCKED” <ip-addr of client to be blocked>. If the Ip address is not a known ip address it will throw an exception



```
Grading for: blocked ...
5.0
```

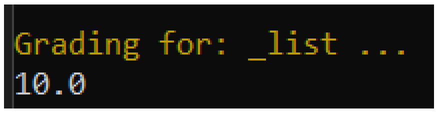
### [EVENT]: Message Relayed

It is the confirmation log printed at the server that it has received a message from the sender and delivered to the receiver.

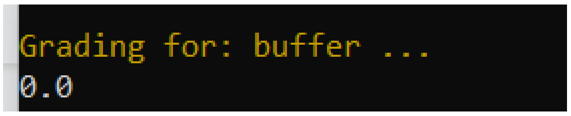
## 5 - Command/Event for Client

### [17.0] LOGIN <server-ip> <server-port> + Exception Handling

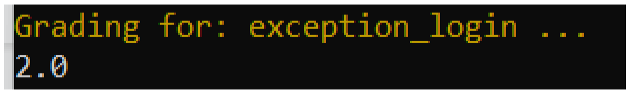
The client needs to login to the server to send messages or interact with other clients. To do so he will type LOGIN <server-ip> <server-port-no>. After this the server will add the client to the list of logged in clients. This list is kept in a structure which is a member of a global class extended by both server and client. In case if the server ip or server-port is not correct it will throw an exception.



```
Grading for: _list ...
10.0
```



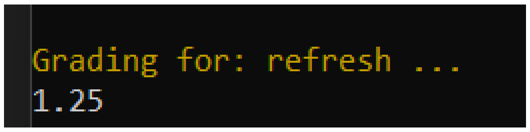
```
Grading for: buffer ...
0.0
```



```
Grading for: exception_login ...
2.0
```

## [5.0] REFRESH

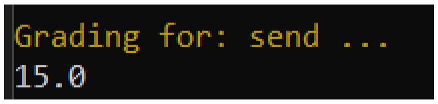
This command is used by the client at a time to get an updated list of the users which are logged in. It will fetch the updated logged in client list from the server.



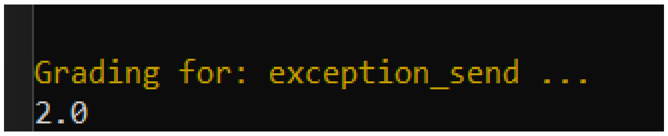
```
Grading for: refresh ...
1.25
```

## [17.0] SEND <client-ip> <msg> + Exception Handling

This command is used by a client to send a text message to another client with the ip address of the destination client. This command makes use of the server to pass the message and hence is not peer to peer. The command format is SEND <client-ip> <msg>, the send score is 15 in our implementation and exception handling of send is 2.



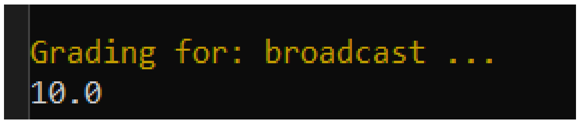
```
Grading for: send ...
15.0
```



```
Grading for: exception_send ...
2.0
```

## [10.0] BROADCAST <msg>

This functionality will iterate over a list of logged in clients and will send the message to all of the clients via server except the clients who have blocked the current sender.

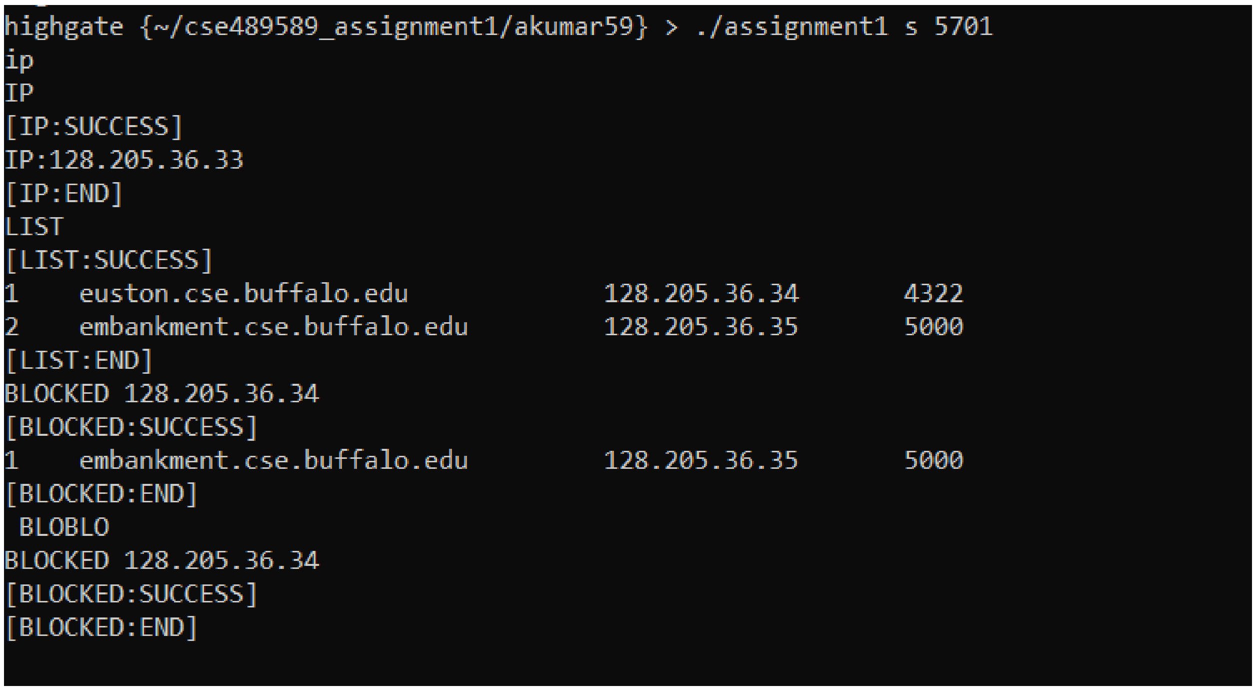


```
Grading for: broadcast ...
10.0
```

## **[7.0]** BLOCK <client-ip> + Exception Handling

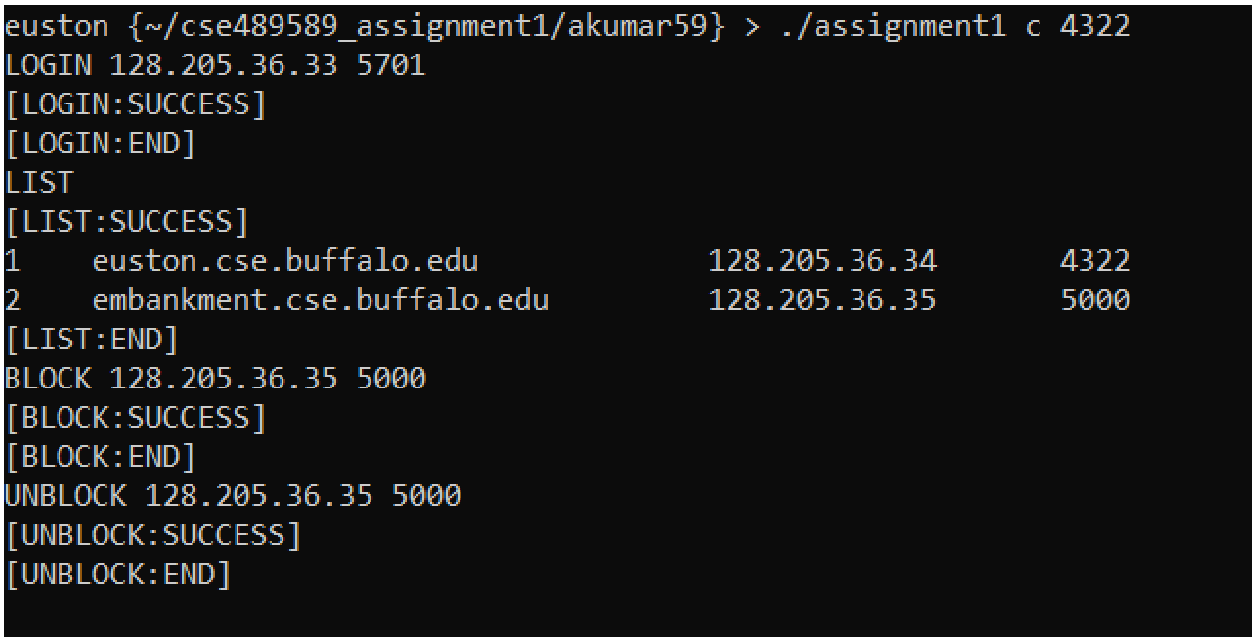
A client or server runs this command with the ip address to add a client to its block list. If the client is not logged-in or if the client is not valid it will though an exception
The grader showed 0.0, but on running the code on terminal it worked as expected as shown below.

Client 1:



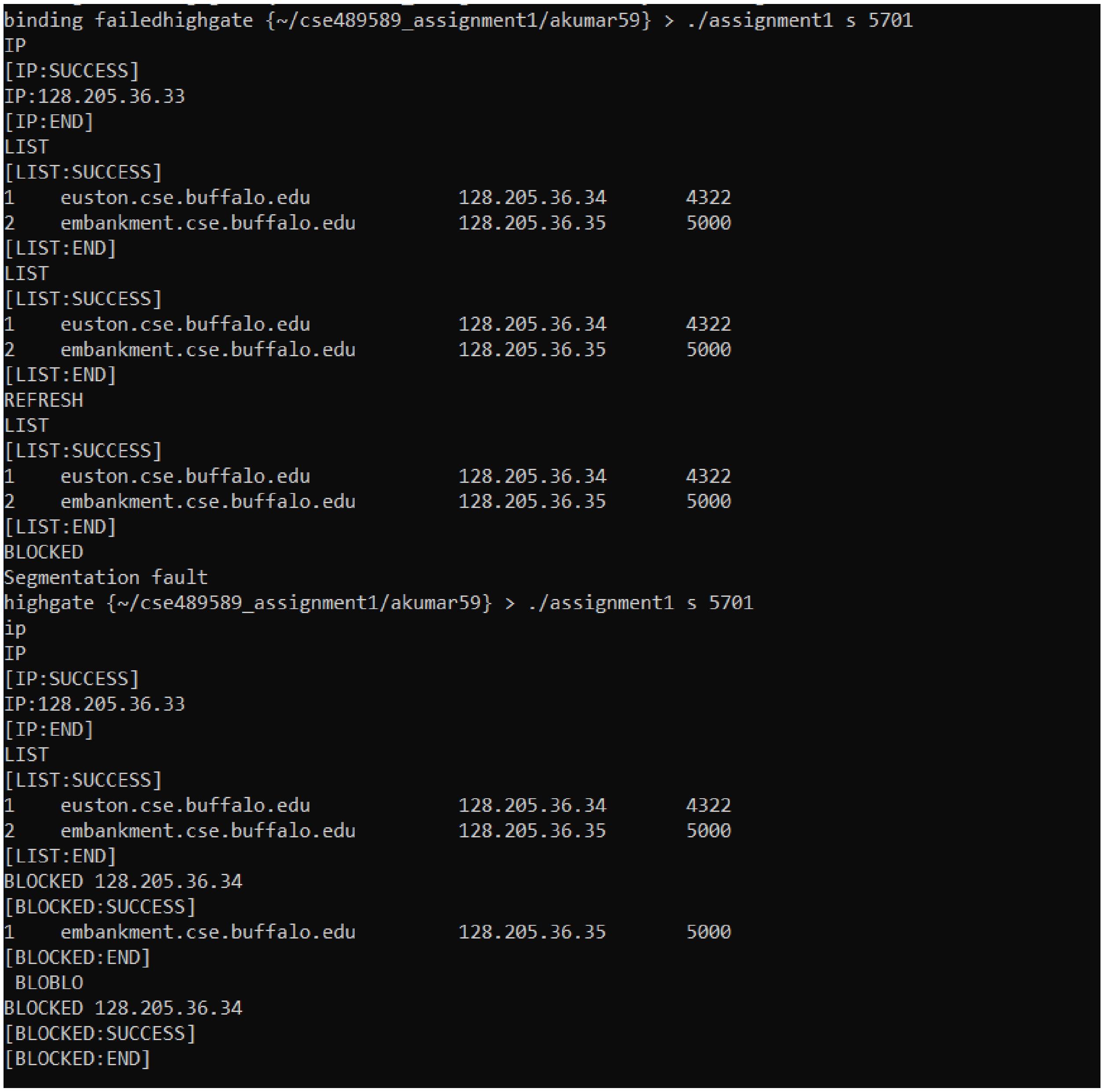
```
highgate {~/cse489589_assignment1/akumar59} > ./assignment1 s 5701
ip
IP
[IP:SUCCESS]
IP:128.205.36.33
[IP:END]
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
BLOCKED 128.205.36.34
[BLOCKED:SUCCESS]
1    embankment.cse.buffalo.edu        128.205.36.35       5000
[BLOCKED:END]
 BLOBLO
BLOCKED 128.205.36.34
[BLOCKED:SUCCESS]
[BLOCKED:END]
```

Client 2:

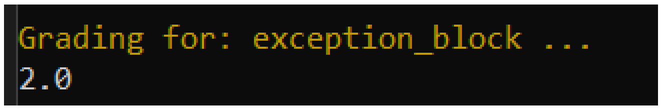


```
euston {~/cse489589_assignment1/akumar59} > ./assignment1 c 4322
LOGIN 128.205.36.33 5701
[LOGIN:SUCCESS]
[LOGIN:END]
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
BLOCK 128.205.36.35 5000
[BLOCK:SUCCESS]
[BLOCK:END]
UNBLOCK 128.205.36.35 5000
[UNBLOCK:SUCCESS]
[UNBLOCK:END]
```

Server:



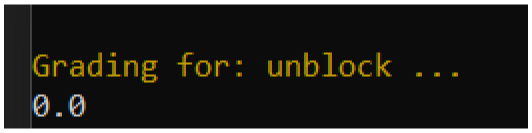
```
binding failedhighgate {~/cse489589_assignment1/akumar59} > ./assignment1 s 5701
IP
[IP:SUCCESS]
IP:128.205.36.33
[IP:END]
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
REFRESH
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
BLOCKED
Segmentation fault
highgate {~/cse489589_assignment1/akumar59} > ./assignment1 s 5701
ip
IP
[IP:SUCCESS]
IP:128.205.36.33
[IP:END]
LIST
[LIST:SUCCESS]
1    euston.cse.buffalo.edu            128.205.36.34       4322
2    embankment.cse.buffalo.edu        128.205.36.35       5000
[LIST:END]
BLOCKED 128.205.36.34
[BLOCKED:SUCCESS]
1    embankment.cse.buffalo.edu        128.205.36.35       5000
[BLOCKED:END]
 BLOBLO
BLOCKED 128.205.36.34
[BLOCKED:SUCCESS]
[BLOCKED:END]
```



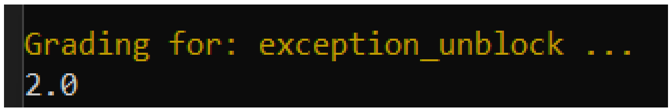
```
Grading for: exception_block ...
2.0
```

## [4.5] UNBLOCK <client-ip> + Exception Handling

This functionality allows a client or server to remove a client from its block list. If the client is not a valid ip address logged in it will throw an exception.



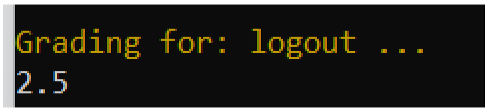
```
Grading for: unblock ...
0.0
```



```
Grading for: exception_unblock ...
2.0
```

## [2.5] LOGOUT

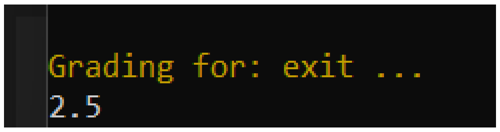
Logout functionality will log out the current logged in client. He will be removed from the list of logged_in clients. However he can still use ip, author and other basic commands.



```
Grading for: logout ...
2.5
```

## [2.5] EXIT

Exit will log out the client from the server and terminate the client session. Unlike logout in exit the client can’t issue any commands if he has logged out



```
Grading for: exit ...
2.5
```

## [EVENT]: Message Received

On receiving a message a client will  print the message on the screen in the required format.