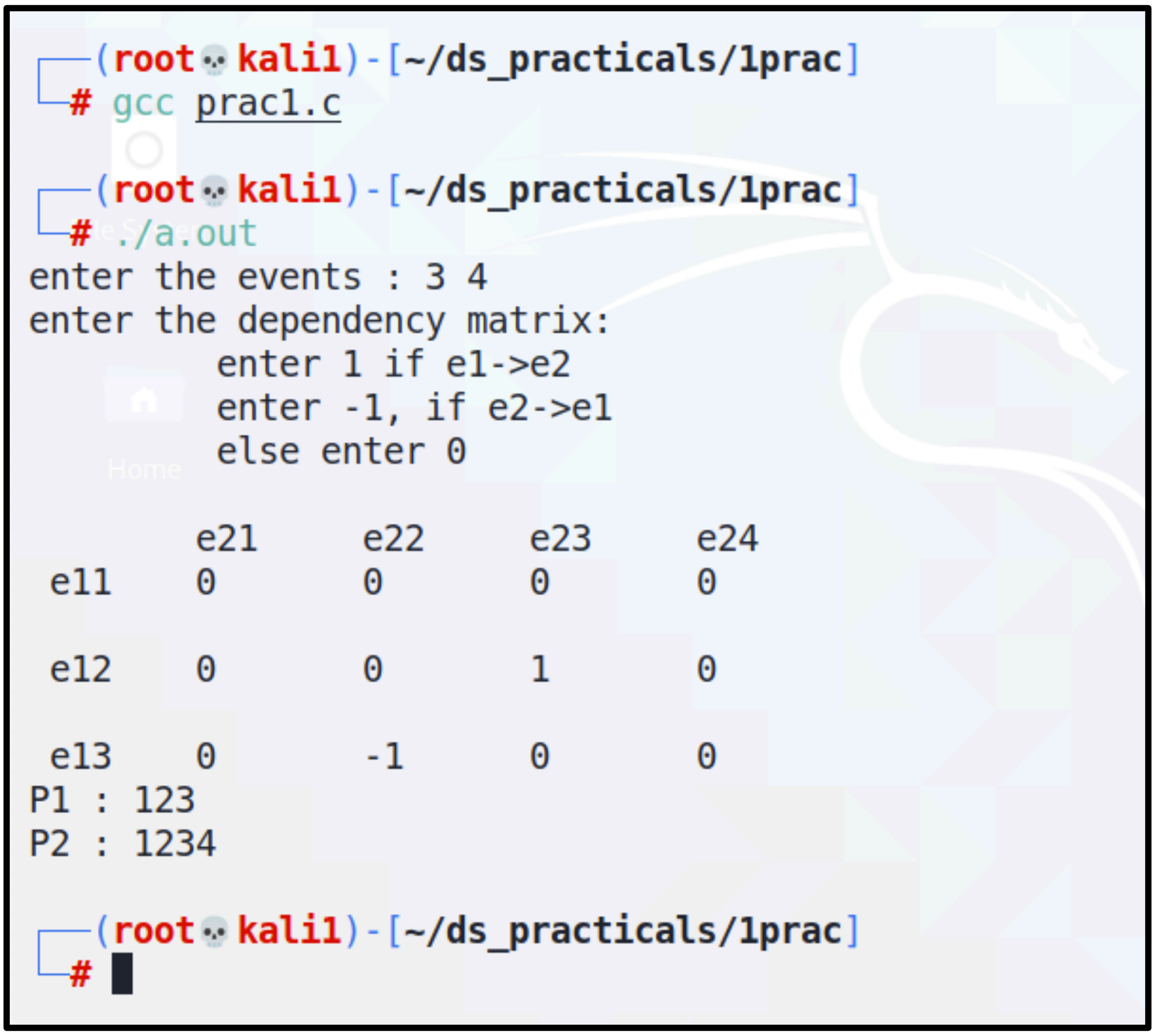
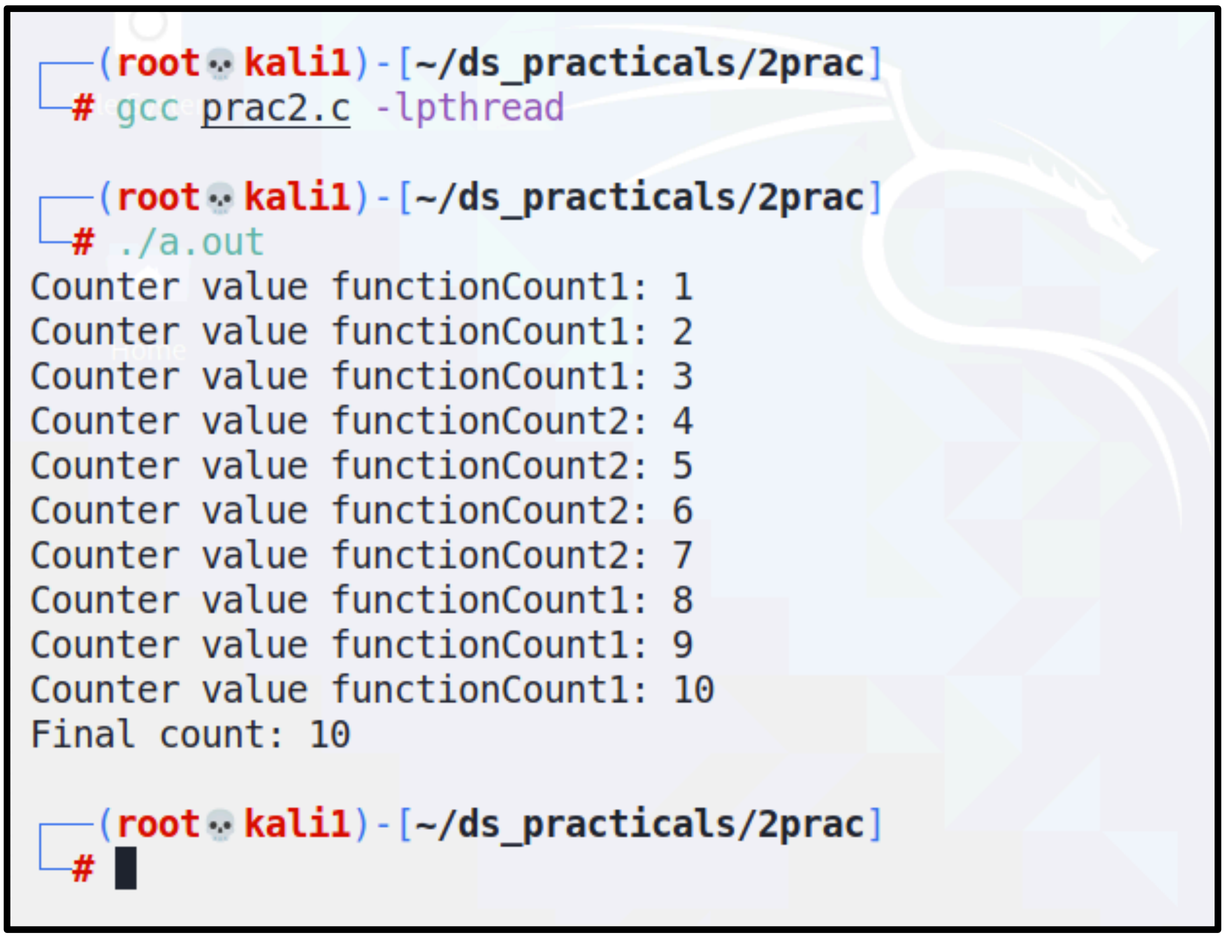
Practical 1:



```
┌──(root💀kali1)-[~/ds_practicals/1prac]
└─# gcc prac1.c

┌──(root💀kali1)-[~/ds_practicals/1prac]
└─# ./a.out
enter the events : 3 4
enter the dependency matrix:
         enter 1 if e1->e2
         enter -1, if e2->e1
         else enter 0

        e21     e22     e23     e24
 e11    0       0       0       0

 e12    0       0       1       0

 e13    0       -1      0       0
P1 : 123
P2 : 1234

┌──(root💀kali1)-[~/ds_practicals/1prac]
└─# █
```

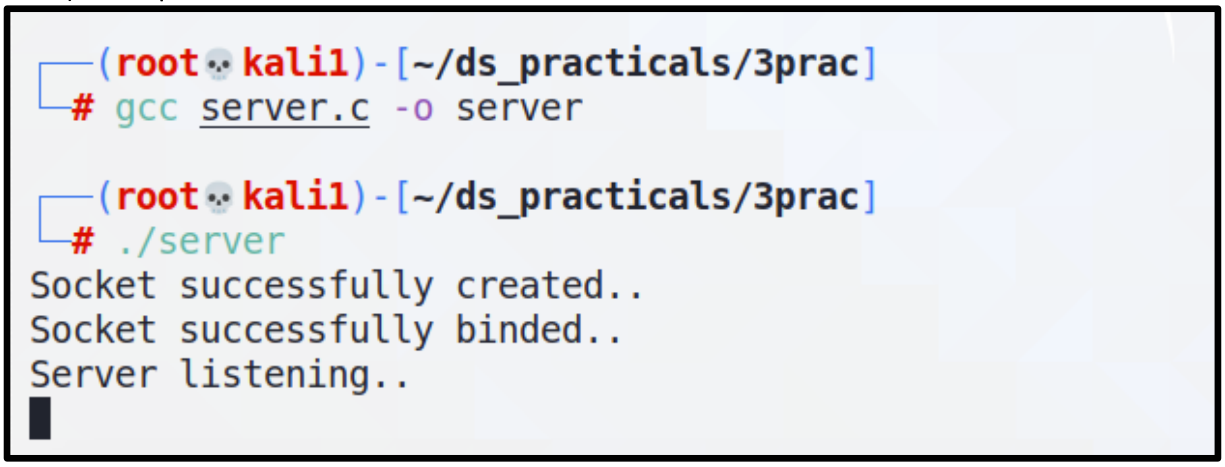
Practical 2:



```
┌──(root💀kali1)-[~/ds_practicals/2prac]
└─# gcc prac2.c -lpthread

┌──(root💀kali1)-[~/ds_practicals/2prac]
└─# ./a.out
Counter value functionCount1: 1
Counter value functionCount1: 2
Counter value functionCount1: 3
Counter value functionCount2: 4
Counter value functionCount2: 5
Counter value functionCount2: 6
Counter value functionCount2: 7
Counter value functionCount1: 8
Counter value functionCount1: 9
Counter value functionCount1: 10
Final count: 10

┌──(root💀kali1)-[~/ds_practicals/2prac]
└─# 
```

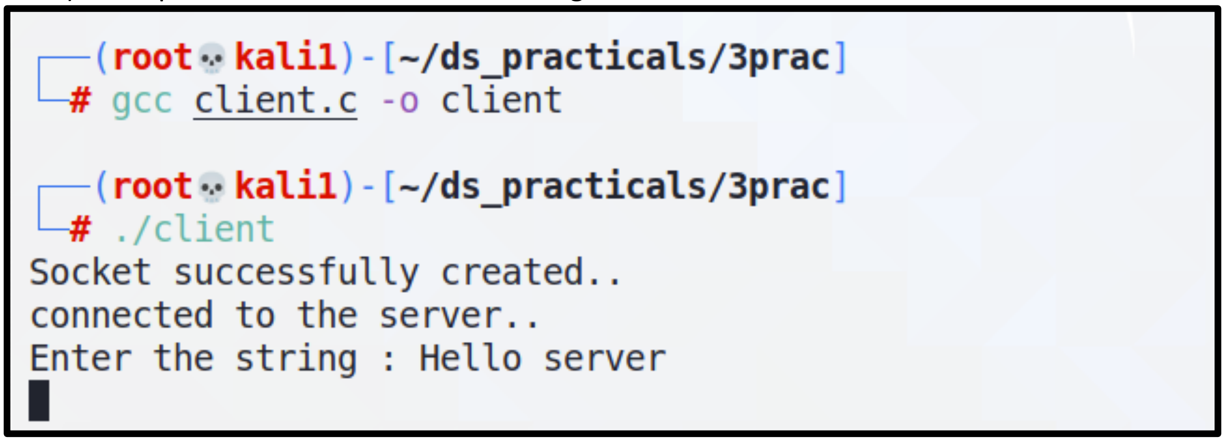
Practical 3:

1) Compile server code and listen:



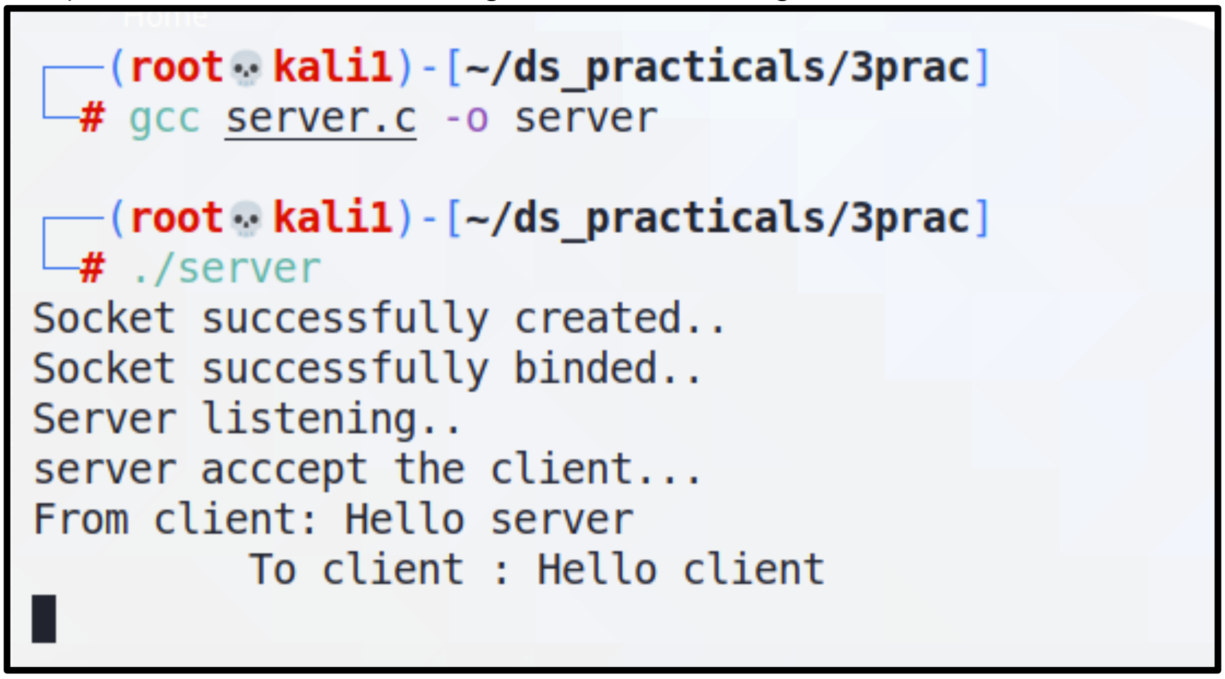
```
┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# gcc server.c -o server

┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# ./server
Socket successfully created..
Socket successfully binded..
Server listening..
█
```

2) Compile client code and send message to server:



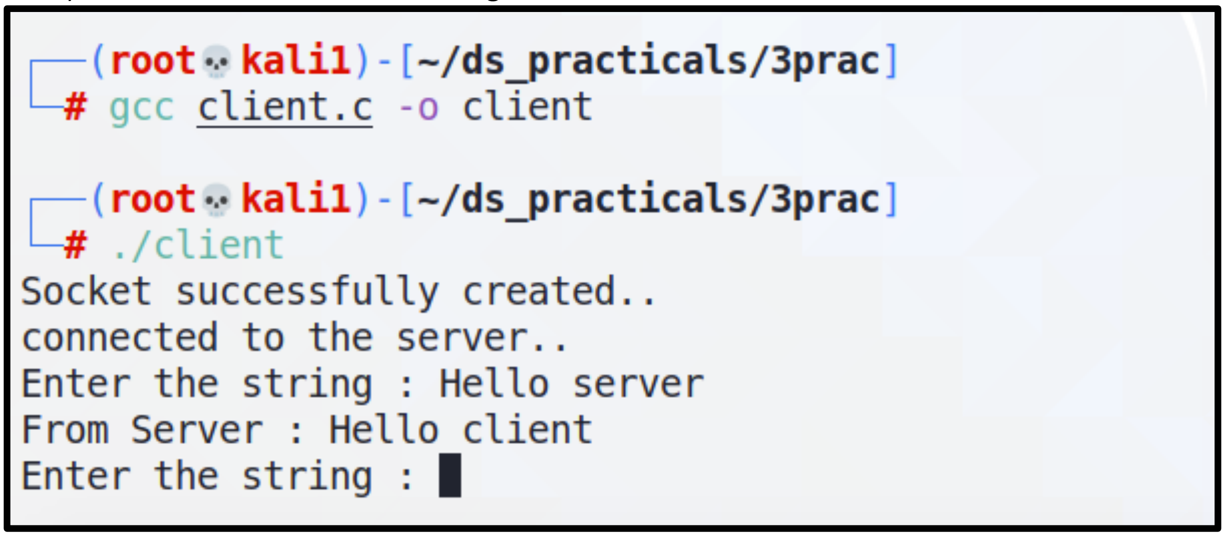
```
┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# gcc client.c -o client

┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# ./client
Socket successfully created..
connected to the server..
Enter the string : Hello server
█
```

3) Server receives client's message. Also, send a message from server to client:



```
┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# gcc server.c -o server

┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: Hello server
         To client : Hello client
█
```

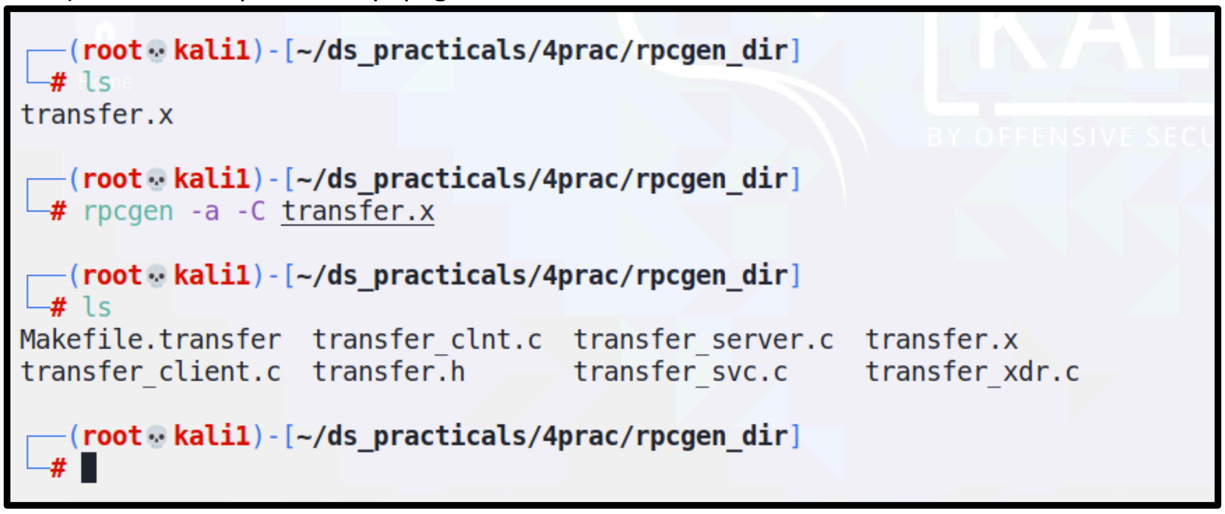
4) Client receives server’s message:



```
┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# gcc client.c -o client

┌──(root💀kali1)-[~/ds_practicals/3prac]
└─# ./client
Socket successfully created..
connected to the server..
Enter the string : Hello server
From Server : Hello client
Enter the string : █
```

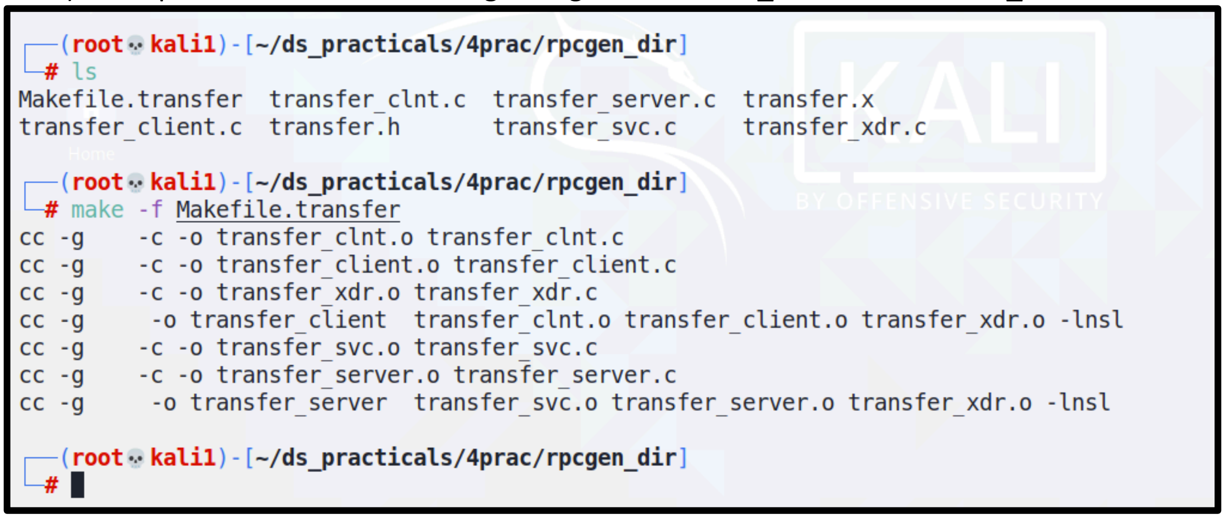
Practical 4:

1) Generate rpc stubs by rpcgen:



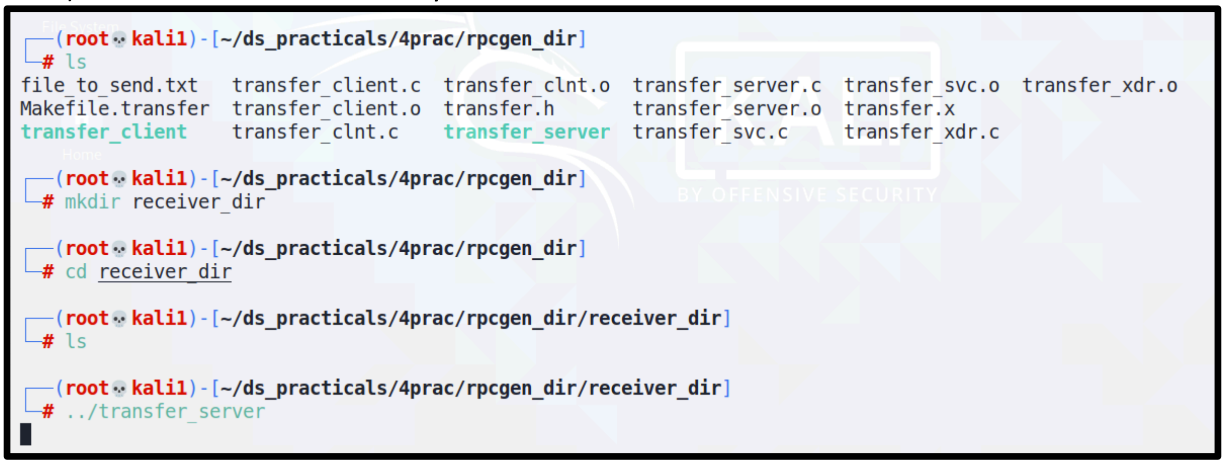
```
┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ls
transfer.x

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# rpcgen -a -C transfer.x

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ls
Makefile.transfer  transfer_clnt.c  transfer_server.c  transfer.x
transfer_client.c  transfer.h       transfer_svc.c     transfer_xdr.c

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# █
```

2) Compile the code after making changes to transfer_client.c & transfer_server.c:



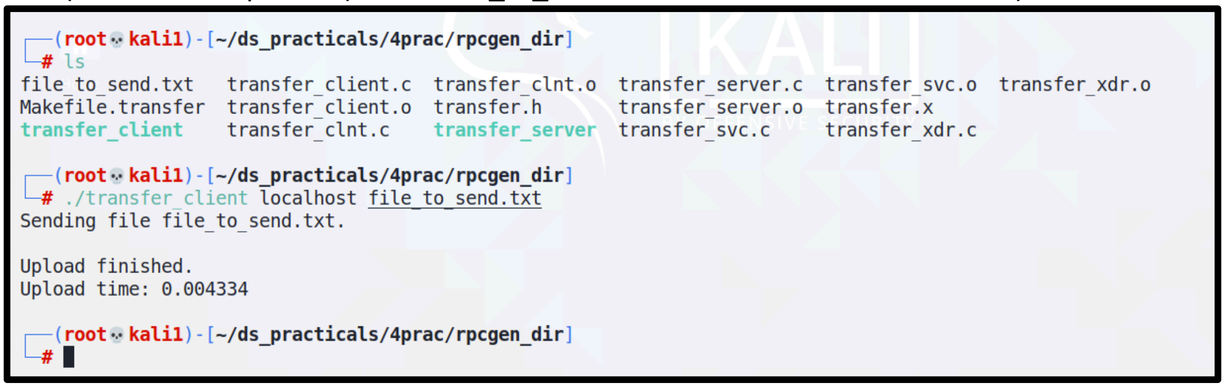
```
┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ls
Makefile.transfer  transfer_clnt.c  transfer_server.c  transfer.x
transfer_client.c  transfer.h       transfer_svc.c     transfer_xdr.c

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# make -f Makefile.transfer
cc -g    -c -o transfer_clnt.o transfer_clnt.c
cc -g    -c -o transfer_client.o transfer_client.c
cc -g    -c -o transfer_xdr.o transfer_xdr.c
cc -g     -o transfer_client  transfer_clnt.o transfer_client.o transfer_xdr.o -lnsl
cc -g    -c -o transfer_svc.o transfer_svc.c
cc -g    -c -o transfer_server.o transfer_server.c
cc -g     -o transfer_server  transfer_svc.o transfer_server.o transfer_xdr.o -lnsl

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# █
```

3) Create a receiver directory and start server:



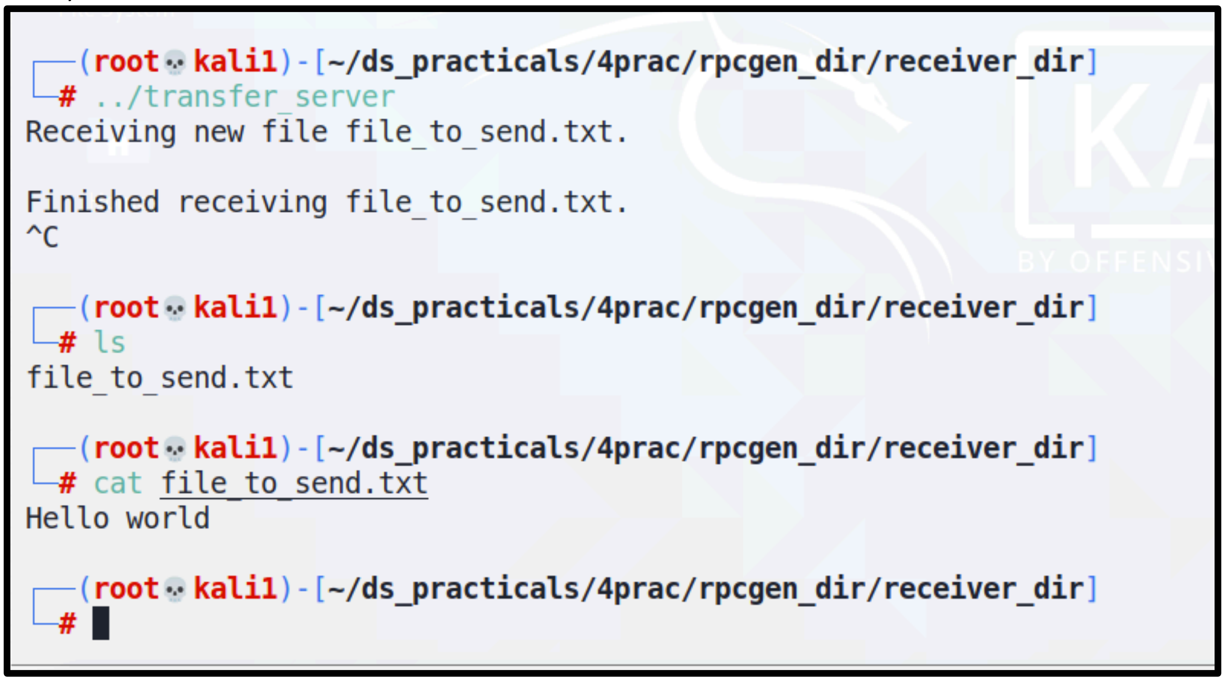
```
┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ls
file_to_send.txt   transfer_client.c  transfer_clnt.o    transfer_server.c  transfer_svc.o  transfer_xdr.o
Makefile.transfer  transfer_client.o  transfer.h         transfer_server.o  transfer.x
transfer_client    transfer_clnt.c    transfer_server    transfer_svc.c     transfer_xdr.c

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# mkdir receiver_dir

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# cd receiver_dir

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# ls

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# ../transfer_server
█
```

4) Send a sample file (named file_to_send.txt and contents “Hello world”):



```
┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ls
file_to_send.txt   transfer_client.c  transfer_clnt.o   transfer_server.c  transfer_svc.o  transfer_xdr.o
Makefile.transfer  transfer_client.o  transfer.h        transfer_server.o  transfer.x
transfer_client    transfer_clnt.c    transfer_server   transfer_svc.c     transfer_xdr.c

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# ./transfer_client localhost file_to_send.txt
Sending file file_to_send.txt.

Upload finished.
Upload time: 0.004334

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir]
└─# 
```

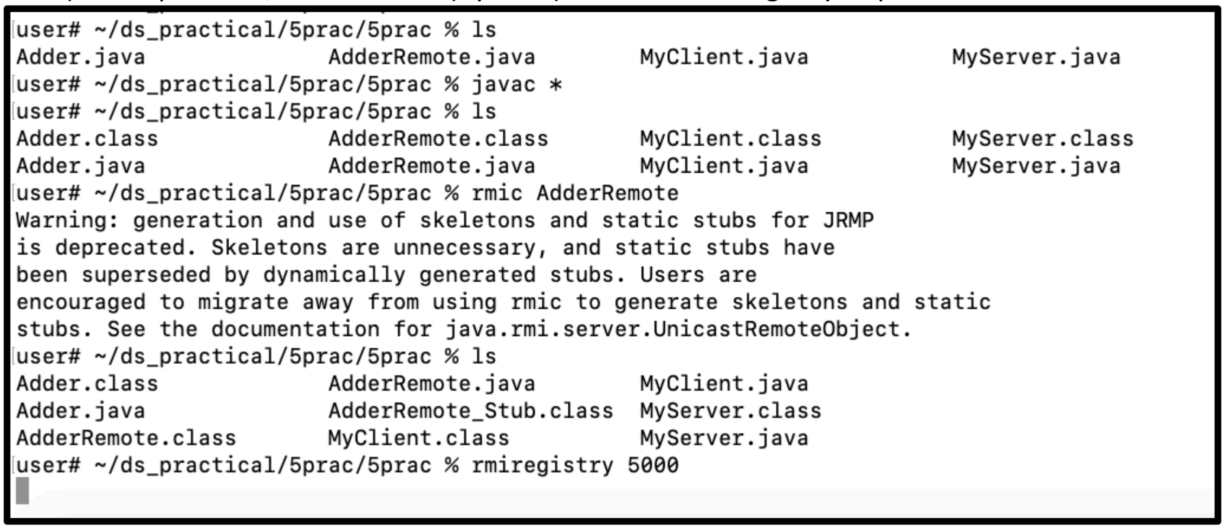
5) Server receives the file:



```
┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# ../transfer_server
Receiving new file file_to_send.txt.

Finished receiving file_to_send.txt.
^C

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# ls
file_to_send.txt

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# cat file_to_send.txt
Hello world

┌──(root💀kali1)-[~/ds_practicals/4prac/rpcgen_dir/receiver_dir]
└─# 
```

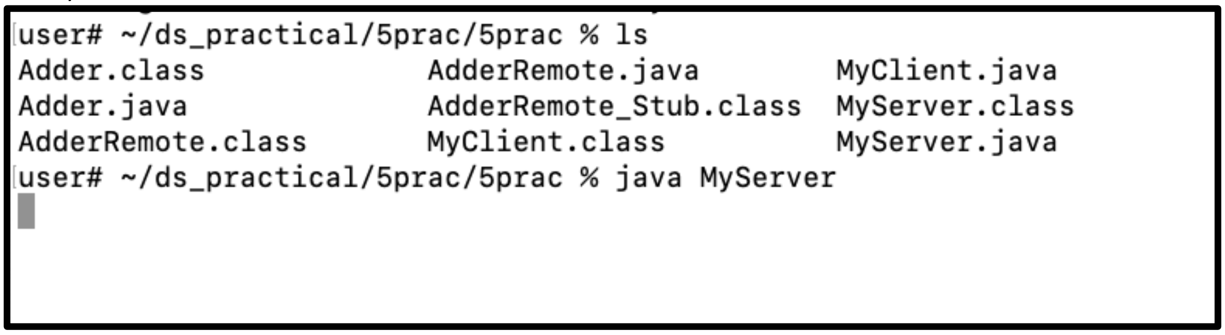
Practical 5:

1) Compile files, create stubs (by rmic) and start rmiregistry at port 5000:



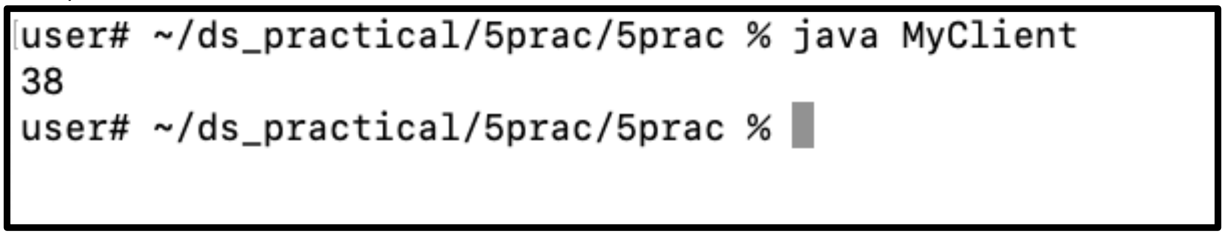
```
user# ~/ds_practical/5prac/5prac % ls
Adder.java              AdderRemote.java        MyClient.java           MyServer.java
user# ~/ds_practical/5prac/5prac % javac *
user# ~/ds_practical/5prac/5prac % ls
Adder.class             AdderRemote.class       MyClient.class          MyServer.class
Adder.java              AdderRemote.java        MyClient.java           MyServer.java
user# ~/ds_practical/5prac/5prac % rmic AdderRemote
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
user# ~/ds_practical/5prac/5prac % ls
Adder.class             AdderRemote.java        MyClient.java
Adder.java              AdderRemote_Stub.class  MyServer.class
AdderRemote.class       MyClient.class          MyServer.java
user# ~/ds_practical/5prac/5prac % rmiregistry 5000
```

2) Start server in one tab:



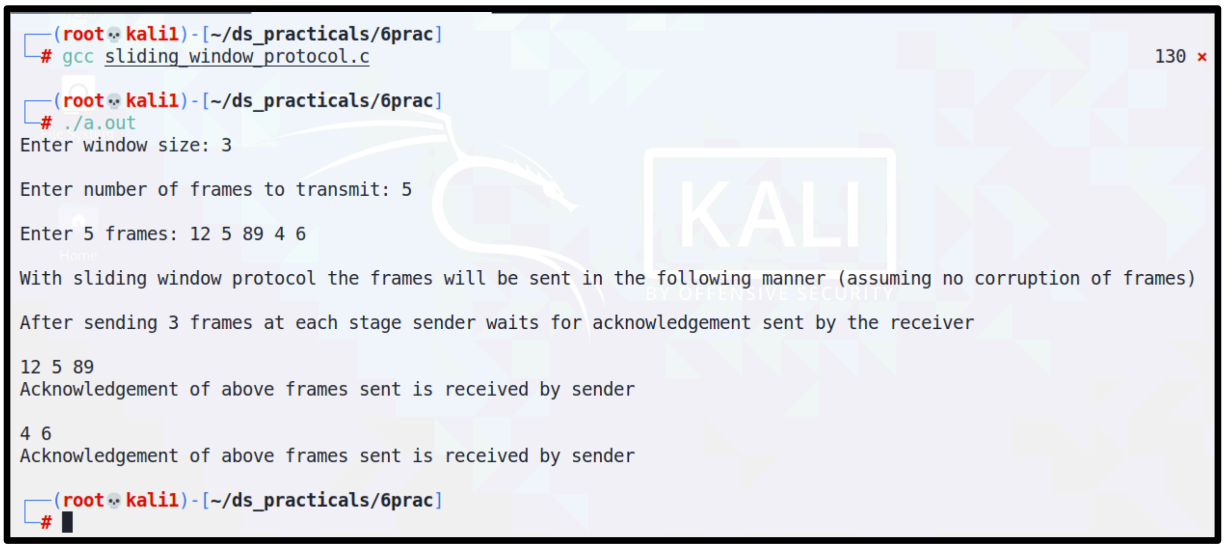
```
user# ~/ds_practical/5prac/5prac % ls
Adder.class             AdderRemote.java        MyClient.java
Adder.java              AdderRemote_Stub.class  MyServer.class
AdderRemote.class       MyClient.class          MyServer.java
user# ~/ds_practical/5prac/5prac % java MyServer
```

3) Execute client in another tab:



```
user# ~/ds_practical/5prac/5prac % java MyClient
38
user# ~/ds_practical/5prac/5prac %
```

Practical 6:



```
┌──(root💀kali1)-[~/ds_practicals/6prac]
└─# gcc sliding_window_protocol.c                                                    130 ×

┌──(root💀kali1)-[~/ds_practicals/6prac]
└─# ./a.out
Enter window size: 3

Enter number of frames to transmit: 5

Enter 5 frames: 12 5 89 4 6

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender

┌──(root💀kali1)-[~/ds_practicals/6prac]
└─# █
```