CSC 573
Fall 2012

Compile and Run Instructions for:
Project 1 and Project 2 and Extra Credit

Name 1: Altaf Hussain Palanawala Student ID1: 001029804
Name 2: Raheel Kazi Student ID2: 001079586

Project 1-Peer to Peer System

Project 1 consists of a simple Peer to Peer File Transfer System with centralized index.

The folder consists of the following files:

1. ServerMC.java – This is the main Server which holds all the data about the clients
2. ClientHandler.java – This is again present on the server side to assist the server to handle the clients. Thus as new clients join in the server spawns new threads, which are looked after by this class.
3. Client2.java – This is the client file.
4. ClientServe.java – This file is on the client side and it allows the client to listen on an upload port while it performs its own functions.
5. ClientServeHandler.java – This is similar to ClientHandler.java. However instead of assisting the server this file assists the client.

Initial Setup
Create some RFC files on each of the client side.
The RFC files should have a name of the format "RFCFile123.txt" or "RFCFile345.txt" .. In general "RFCFile(RFC number).txt".

Running the Peer to Peer System

1. Open terminals on each of the machines.
2. On any one terminal which is needed as the server compile the Server code as such :
   `javac ServerMC.java`

3. Run the server by `java ServerMC`
4. Similarly now run each of the client side by first compiling each of the client files: `javac client2.java`

5. Run the Client file as such `java Client2 "serverIP"`
6. Note: When the server starts it displays its IP so use the same IP address.
7. After the server and client are connected the instructions on the terminal can be followed which are self-explanatory.
8. When the client needs to exit just type a bye on his terminal.

Project 2: Go Back N ARQ

1. FTPClient.java
2. FTPServer.java

To compile and run the above programs do the following:

/* For Server (Receiver) */

```
javac   FTPServer.java
java     FTPServer   7735   outputFile   p
```

where outputFile is the file wherein the data from the sender will be copied, p is the loss probability and 7735 is the port on which the server is running

/* For Client (Sender) */

```
javac   FTPClient.java
java FTPClient server-hostname 7735 InputFile N MSS
```

where InputFile is the file you want to transfer, server-hostname is the IP address of the server, 7735 is the port on which the server is listening, N is the window size and MSS is the maximum segment size.

Run the Server first and then the client.
When the client runs, the following statement is printed:

```
SEND(ftp or FTP to transfer file) ...
```

After this it waits for the "ftp" command to be send. Just type ftp and hit enter.

```
ftp
```

The sender (client) then starts sending the file and the receiver then starts receiving it.

Extra Credit Selective Repeat

1. SelectiveRepeatClient.java
2. SelectiveRepeatServer.java

To compile and run the above programs do the following:

/* For Server (Receiver) */

```
javac   SelectiveRepeatServer.java
java   SelectiveRepeatServer  7735   outputFile   p
```

where outputFile is the file wherein the data from the sender will be copied, p is the loss probability and 7735 is the port on which the server is running

/* For Client (Sender) */

```
javac   SelectiveRepeatClient.java
java SelectiveRepeatClient server-hostname 7735
InputFile N MSS
```

where InputFile is the file you want to transfer, server-hostname is the IP address of the server, 7735 is the port on which the server is listening, N is the window size and MSS is the maximum segment size.

Run the Server first and then the client.
When the client runs, the following statement is printed:

```
SEND(ftp or FTP to transfer file) ...
```

After this it waits for the "ftp" command to be send. Just type ftp and hit enter.

```
ftp
```

The sender (client) then starts sending the file and the receiver then starts receiving it.