Computer Science 390
Project 1 - A Shared Memory Chat Program

Due: Mon., Mar. 6, 11:59 p.m. You may work in groups of two if you wish.

For this project, you are to write two programs in C that communicate with each other using POSIX shared memory. The programs should implement a simple chat feature. They must coordinate their communication using semaphores. The source code for the two programs should be stored in files named "chat0.c" and "chat1.c".

In your implementation, chat0 must

- set up the shared memory segment; and
- initialize the semaphores that the two processes will use to communicate.

The shared memory segment must be deleted at the termination of a chat session.

When writing your program, you should assume that chat0 begins to execute before chat1, and that chat0 sends the first message. The two programs should communicate in a see-saw manner. That is, chat0 will read a line of text from the user, place it in the shared memory, and then wait for and print a reply from chat1 before reading any additional text from the user. Similarly, chat1 will wait for text to appear in shared memory and print it out before reading any text from its user.

The programs should clean up and terminate whenever any user sends the string "END".

Here is an example of what the output of your programs should look like when they are executed:

```
matthews@sand:~/shared_memory$ ./chat0
Enter some text to send to 1: I say foo.
chat1: I say bar.
Enter some text to send: END

matthews@sand:~/shared_memory$ ./chat1
chat0: I say foo.
Enter some text to send to 0: I say bar.
chat0: END
```

What to turn in: When you have finished with your programs, use the tar command to package the source.[1] Do not include anything other then source files (.c and .h) and a Makefile (not required) in your tarball.

Send me an email message with the tarball as an attachment. Make sure the message has the subject "CS 390 Project 1 <Your last name >'

Finally, print out a hard copy of the source, and assemble it as header file(s) (if any), chat0.c, chat1.c. Staple everything together. You must submit your hard copy no later than the next day after your electronic submission to receive full credit.

[1] If you are unfamiliar with tar, the shell command `tar -cvf my_project.tar source0.c source1.c source2.c` will create a tarball that contains the listed source files. You can use `tar -tf my_project.tar` to verify the contents of the tarball after you have created it.